POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Analisi e gestione dell'attestazione remota in reti virtuali

Relatori
Prof. Antonio Lioy
Dott. Marco De Benedictis

Candidato
Salvatore Oddo

Anno accademico 2017-2018

Indice

Capitolo 1

Introduzione

Le reti di telecomunicazioni si sono evolute nel corso degli anni. Inizialmente erano presenti delle reti dedicate ad uno specifico servizio, ad esempio la rete telefonica veniva utilizzata solo per effettuare delle chiamate. Negli anni successivi si sviluppano nuovi servizi che sfruttano le risorse fisiche già esistenti. L'innovazione e l'evoluzione tecnologica ci ha portato ad avere l'interconnessione di una grande varietà di dispositivi che possono trovarsi in luoghi differenti e possono spostarsi mentre comunicano. I fornitori di servizi si stanno adattando a questo approccio supportando la mobilità e fornendo l'accesso a numerose applicazioni, ciò ha portato i meccanismi di virtualizzazione a diventare un aspetto strategico di primaria importanza per le aziende. La riduzione dei costi, l'ottimizzazione delle risorse ed una maggiore flessibilità sono i motivi che hanno portato all'adozione delle reti virtuali ed allo sviluppo di uno standard che permettesse l'interoperabilità di hardware prodotto da aziende differenti. La possibilità di accogliere le richieste che arrivano variando dinamicamente le risorse a disposizione permette di soddisfare anche picchi di carico senza l'obbligo di realizzare un'infrastruttura fisica sovradimensionata che in molti casi sarebbe sottoutilizzata.

L'esigenza di accedere a funzionalità sempre diverse e nuove ha portato le aziende ad investire in tecnologie che potessero offrire almeno lo stesso livello di prestazioni e sicurezza delle reti tradizionali ma che riducessero i costi necessari all'implementazione ed alla gestione delle nuove funzioni, ciò ha permesso lo sviluppo delle reti virtuali. Gli ultimi anni hanno visto un incremento delle tecnologie cloud sia pubblico sia privato. Nel cloud pubblico le risorse sono gestite ed appartengono al fornitore di servizi. Nel cloud privato le aziende posseggono le risorse all'interno dei propri data center. La Figura 1.1 estratta dall'ultimo report sullo stato del cloud di RightScale [1], nota

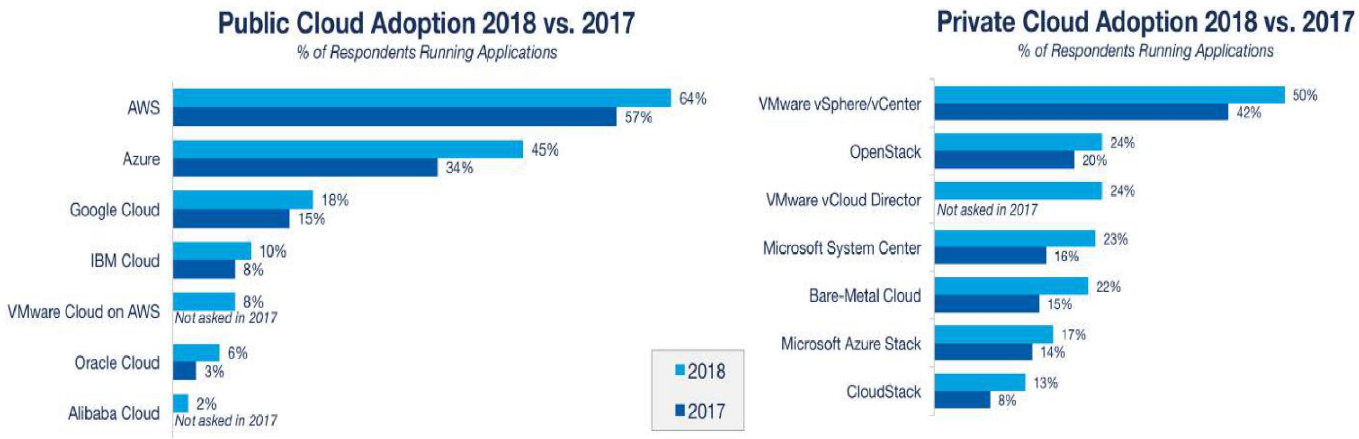

Figura 1.1. Adozione di cloud pubblico e privato anni 2017 e 2018 (fonte:[2]).

azienda che offre servizi di tipo SaaS (Software as a Service), mostra l'incremento dell'utilizzo delle soluzioni cloud sia pubbliche sia private fra l'anno 2017 e il 2018. I principali fornitori di servizi

cloud hanno visto un aumento delle richieste di utilizzo dei loro servizi, inoltre si può notare come la somma totale superi il 100% sinonimo che le aziende tendono ad utilizzare contemporaneamente più soluzioni cloud. Possiamo dedurre che il settore del cloud computing è in forte espansione e le aziende stanno già trasformando la loro infrastruttura tradizionale adattandola al paradigma offerto dal cloud o ricorrendo direttamente a soluzioni esterne alla propria infrastruttura.

Un altro aspetto da tenere in considerazione sono le principali sfide e preoccupazioni che è necessario affrontare per utilizzare il cloud. Come è possibile vedere dalla Figura 1.2 la principale

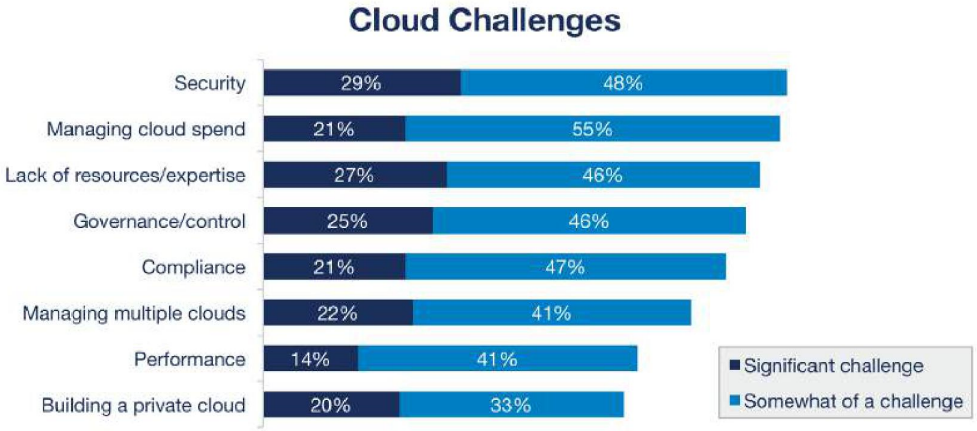

Figura 1.2. Principali sfide del cloud (fonte:[2]).

sfida è legata allo sviluppo di soluzioni cloud tenendo in considerazione i problemi di sicurezza che si possono verificare. I dati degli utenti potrebbero essere localizzati ovunque e per questo motivo è necessario adottare dei meccanismi che permettano di rispettare le principali proprietà legate alla sicurezza come la necessità di rilevare delle modifiche che potrebbero essere apportate ai dati, l'impossibilità di visualizzare i dati per chi non è autorizzato a farlo, la disponibilità dei dati in qualsiasi momento e attraverso qualsiasi dispositivo.

Per supportare questo scenario che prevede l'utilizzo di servizi sicuri su reti virtuali è stato sviluppato il paradigma Network Function Virtualization (NFV) come parte dello European Telecommunications Standard Institute (ETSI). In seguito sono stati aggiunti numerosi gruppi di lavoro ognuno dei quali si occupa di un'area specifica per facilitare lo sviluppo di soluzioni interoperabili ed aperte attraverso la formulazione di standard che le aziende devono seguire. Fra questi gruppi è presente anche quello denominato NFV SEC che si occupa di tutte le tematiche inerenti la sicurezza.

Il lavoro di tesi si inserisce nell'ambito della sicurezza di una rete NFV. La necessità di sapere che i propri dati attraverseranno delle reti virtuali sicure ed affidabili e che le computazioni avverranno su server fidati ovunque essi si trovino ha portato alla realizzazione di un'architettura che offre un'infrastruttura virtualizzata ed una modalità di verifica dello stato di fiducia legato ai server fisici.

La tesi, dopo questa breve introduzione, analizza la tecnologia nota come Trusted Computing, introduce le caratteristiche del Trusted Platform Module e descrive la modalità di funzionamento dell'attestazione remota nel Capitolo 2. Verranno introdotte le caratteristiche delle reti virtuali e dello standard ETSI NFV nel Capitolo 3. Il Capitolo 4 descriverà ad alto livello lo scenario, i principali software utilizzati ed espliciterà l'obiettivo della tesi. Il Capitolo 5 illustrerà l'architettura realizzata, i componenti che è stato necessario estendere ed il flusso di esecuzione della soluzione realizzata. Il Capitolo 6 spiegherà i dettagli implementativi della soluzione. Il Capitolo 7 mostrerà i test effettuati ed i risultati ottenuti. Il Capitolo 8 illustrerà le conclusioni ed i possibili sviluppi futuri. Infine verranno descritte le operazioni per installare e configurare i vari software nel manuale utente in Appendice A e le API REST con cui è possibile contattare l'Attestation Server nel manuale dello sviluppatore in Appendice B.

Capitolo 2

Attestazione remota

In questo capitolo introdurremo il Trusted Computing nella Sezione 2.1 dove analizzeremo le varie funzionalità offerte, i vantaggi e gli svantaggi di tale tecnologia. A seguire tratteremo la struttura architetturale del TPM nella Sezione 2.2 ed infine analizzeremo l'attestazione remota dalla prima all'ultima versione nella Sezione 2.3.

2.1 Il Trusted Computing

Fin dai primi anni 2000 nacque il Trusted Computing Group (TCG), un'associazione costituita da aziende come Intel, IBM, Microsoft, AMD, HP e col passare degli anni se ne aggiunsero molte altre. Il TCG fu il successore di un consorzio nato alla fine degli anni '90 chiamato Trusted Computing Platform Alliance (TCPA) che aveva lo scopo di rendere più sicuri i dispositivi elettronici. Il TCG ereditò le specifiche del chip hardware noto come Trusted Platform Module (TPM) e le rese pubbliche con la versione 1.1; il funzionamento di tale tecnologia che prese il nome di Trusted Computing (TC) si basa su componenti sia hardware sia software ed ancora oggi continua ad essere sviluppata dalle aziende che fanno parte del TCG. Le principali funzionalità offerte dal TC sono il secure boot, il secure I/O, il memory curtaining, il data sealing e la remote attestation [3]. Il secure boot permette di avviare in maniera sicura il sistema basandosi su tecniche di firma digitale. Nella fase di avvio il TPM verifica se il BIOS è stato manipolato ed eventualmente impedisce l'avvio del sistema. Nel caso in cui la precedente verifica vada a buon fine verranno controllati il loader del sistema operativo ed il kernel. Infine le misurazioni cioè gli hash degli eseguibili verranno caricati su appositi registri del TPM. Questa procedura è molto delicata e si basa su un insieme di funzioni sempre fidate dette roots of trust, distinguiamo:

- Root of Trust for Measurement (RTM): permette di realizzare in maniera fidata la misurazione di un elemento del sistema utilizzando un algoritmo di hash;
- Root of Trust for Reporting (RTR): permette di trasmettere in maniera fidata, ad una terza parte, lo stato della piattaforma;
- Root of Trust for Storage (RTS): permette di memorizzare in maniera fidata i dati nella piattaforma o nel TPM grazie ad una gerarchia di chiavi che vede come punto di partenza quella chiamata storage root key (SRK) che sarà memorizzata nel TPM e permetterà di proteggere altre chiavi memorizzate all'esterno del TPM.

Il secure I/O permette di inserire dati nel sistema o di visualizzare dati dal sistema in maniera sicura. Questa funzionalità impedisce che un eventuale malware riesca ad intercettare dati potenzialmente sensibili che transitano sul bus infatti il meccanismo con cui si realizza si basa su tecniche crittografiche per ottenere la proprietà di confidenzialità. Un esempio dell'utilità di questa funzione si può riscontrare nel caso di autenticazione locale eseguita con dati biometrici. Un utente che si

autentica con la propria impronta digitale in un sistema non adeguatamente protetto va incontro a seri rischi per la propria sicurezza.

Il memory curtaining permette di isolare la memoria utilizzata da un certo processo. Esso impedisce ad altri programmi in esecuzione nel sistema di apportare delle modifiche o anche solo di leggere in una zona di memoria che non è stata riservata a loro.

Il data sealing permette di sigillare i dati presenti in memoria. I dati verranno cifrati con algoritmi di cifratura simmetrica ed utilizzeranno una chiave che dipenderà dall'attuale combinazione hardware e software del sistema. Solo con la stessa combinazione di hardware e software sarà possibile generare la chiave per decifrare i dati. Il data sealing è una delle funzionalità più dibattute del TC perchè cambia il modo di intendere i dati. Secondo Ross Anderson, professore di sicurezza informatica all'Università di Cambridge, si passa da un approccio in cui l'utente possiede i dati memorizzati nel proprio elaboratore ad uno in cui i dati restano sotto il controllo del sistema che li ha creati [4]. Questa funzionalità permette di implementare i meccanismi, tanto auspicati dalle grosse aziende produttrici di software e di contenuti, a salvaguardia dei diritti digitali (digital rights management - DRM). I file copiati da un'altra macchina non saranno accessibili perché non si riesce ad avere la stessa combinazione hardware e software necessaria a generare la corretta chiave di decifratura. Solo il software che manipola i dati può prevedere una funzione che permetta di esportarli in maniera non sicura per poterli utilizzare anche in altri elaboratori.

La remote attestation permette di verificare se su un sistema remoto è in esecuzione del software che viene ritenuto fidato. Essa si basa su tecniche di hash per il calcolo delle misure e sulle firme digitali per l'autenticazione e l'integrità dei dati scambiati. Il meccanismo di remote attestation verrà discusso in maniera esaustiva nella Sezione 2.3.

Il TPM è l'elemento principale per il funzionamento di questa tecnologia. Oggi possiamo distinguere vari tipi di TPM che differiscono per modalità realizzative e livelli di sicurezza [5]:

- discrete TPM: è il chip col più alto livello di sicurezza, viene realizzato come dispositivo indipendente e contiene solo le funzionalità descritte dalle specifiche del TCG;
- integrated TPM: ha un livello di sicurezza leggermente inferiore rispetto al precedente perché viene realizzato in un chip che fornisce altre funzionalità oltre quelle di sicurezza;
- firmware TPM: le funzionalità del TPM sono implementate con software eseguito in un ambiente protetto separato dal resto dei programmi in esecuzione su quella CPU;
- software TPM: è un emulatore software delle funzionalità offerte dal TPM e di solito si utilizza solo in ambiente di sviluppo e test; questa è la modalità meno sicura perché viene eseguita nel sistema come un normale programma quindi è vulnerabile sia ad attacchi hardware sia ad attacchi software;
- virtual TPM: è fornito da un hypervisor e permette di realizzare le funzionalità TPM per le macchine virtuali, ognuna delle quali crederà di avere a disposizione un proprio TPM.

Il TCG definisce 5 tipi di credenziali [6] che permettono di fornire solamente le informazioni necessarie per una specifica operazione, ogni credenziale sarà firmata digitalmente da chi la emette:

- Endorsement Credential: è emessa dal costruttore del TPM per certificare la corretta creazione e il corretto inserimento delle endorsement key (EK) nel TPM; questa credenziale conterrà le informazioni relative al costruttore, al modello e alla versione del TPM e la chiave pubblica EK.
- Conformance Credential: è emessa dal costruttore del TPM, dal venditore del TPM o da un'entità esterna ed indica la conformità della piattaforma alle specifiche emesse dal TCG. Si utilizzerà un'unica credenziale per le piattaforme dello stesso tipo.
- Platform Credential: è emessa dal costruttore del TPM, dal venditore del TPM o da un'entità esterna ed indica che una piattaforma contiene un TPM valido e conforme alle specifiche. Essa all'interno conterrà i riferimenti all'Endorsement Credential e alla Conformance Credential.

- Validation Credential: è emessa da un'entità di validazione e contiene le misure di riferimento associate ai componenti presenti nel sistema. Solamente i componenti più sensibili alle minacce di sicurezza verranno misurati (es schede di rete, processori, software, etc) ed il risultato verrà inserito nella credenziale che sarà resa pubblica.

- Identity Credential: è emessa da un servizio fidato che verifica le varie credenziali, essa contiene la chiave pubblica Attestation Identity Key (AIK).

I campi di applicazione del TC sono numerosi e vari. È possibile realizzare delle reti sicure consentendo l'accesso solo ai dispositivi ritenuti fidati. Si può cifrare il disco rigido impedendo che i dati presenti possano essere visualizzati da utenti non autorizzati o su un altro dispositivo, su questo principio si basa la funzionalità BitLocker Drive Encryption che Microsoft ha introdotto da qualche anno. Si può realizzare un sistema DRM anticopia che impedisca la duplicazione non autorizzata dei dati. Si può proteggere l'utente dal phishing eseguendo una verifica sul server che si sta contattando. I principali svantaggi legati all'utilizzo del TC sono l'impossibilità di modificare il software, la perdita del controllo sui propri dati, la difficoltà nel trasferire i dati su un altro pc.

2.2 L'architettura del TPM

Le specifiche pubblicate dal TCG stabiliscono l'architettura del TPM e le funzionalità che esso deve offrire. Ogni TPM è identificato univocamente da una coppia di chiavi asimmetriche dette Endorsement Key (EK). Esse sono generate in fase di produzione del chip direttamente dal costruttore con l'algoritmo RSA a 2048 bit e non possono essere più modificate. La presenza di queste chiavi permette al TPM di dimostrare la propria identità ad una terza parte, in particolare la chiave pubblica viene utilizzata sia nel processo di attestazione remota sia per cifrare dei dati che devono essere trasmessi al chip. I blocchi logici che costituiscono il TPM sono illustrati nella Figura 2.1:

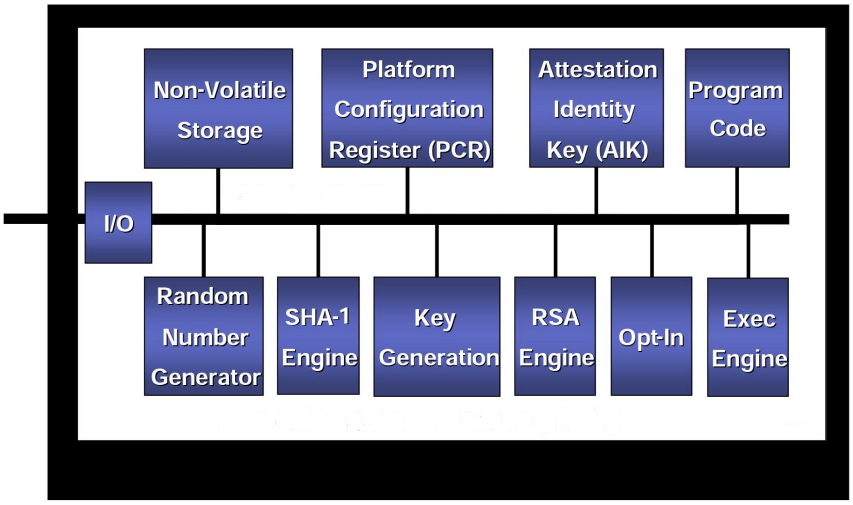

Figura 2.1. Architettura del TPM (fonte:[6]).

- I/O: permette di gestire le comunicazioni in ingresso ed uscita sul bus del chip;

- Non-Volatile Storage: è la memoria persistente che mantiene le informazioni sull'identità del TPM cioè contiene le EK, inoltre su di essa è possibile memorizzare anche la Storage Root Key (SRK) che è la chiave attraverso cui è possibile cifrare i dati da immagazzinare nella memoria del dispositivo;

- Platform Configuration Register (PCR): identifica una serie di registri in cui verranno memorizzate le misurazioni effettuate, in particolare le misurazioni eseguite dopo la fase iniziale dipenderanno sia dal contenuto attuale dei PCR sia dallo stato attuale del sistema. Il TCG specifica che il numero minimo di registri PCR da implementare nel sistema è 16 di cui i primi 8 riservati al TPM e gli altri disponibili per il sistema operativo o le applicazioni;

- Attestation Identity Key (AIK): identifica una coppia di chiavi asimmetriche utilizzate per eseguire l'attestazione remota cioè per dimostrare ad una terza parte che sulla piattaforma gira del software fidato;

- Program Code: contiene il firmware per eseguire le misurazioni;

- Random Number Generator (RNG): è un generatore di numeri pseudo-casuali;

- SHA-1 Engine: è il componente che genera l'hash a 160 bit ricevendo in input il valore attuale dei registri PCR e lo stato attuale del sistema quindi è il motore necessario per eseguire le misurazioni, inoltre permette anche di calcolare l'hash di un file per effettuare la firma digitale;

- Key Generation: è il componente che permette di generare le varie chiavi, sia la coppia di chiavi asimmetriche sia la singola chiave simmetrica;

- RSA Engine: è il componente usato per apporre la firma digitale, per cifrare e decifrare con le storage keys e per decifrare con l'EK;

- Opt-In: è il componente che permette di attivare, disattivare, accendere e spegnere il TPM verificando che l'utente che lo richiede detenga il controllo del TPM;

- Exec Engine: è il motore di esecuzione del codice presente nel blocco Program Code quindi è utilizzato per l'inizializzazione del TPM e per acquisire le misure.

Il TCG, oltre a fornire le specifiche necessarie a realizzare il chip hardware, ha creato una serie di API standard con cui è possibile accedere alle varie funzionalità offerte dal TPM ed esse prendono il nome di Trusted Software Stack (TSS) [7]. Il TSS fornisce anche la traduzione delle politiche imposte dall'utente e dà importanti informazioni sulla protezione della privacy, sulla protezione dei dati sensibili, sulla protezione di partizioni in cui sono presenti i dati di un'applicazione. Un'applicazione può accedere alle funzionalità offerte dal TPM grazie alle interfacce messe a disposizione dal TSS, come si può vedere nella Figura 2.2 si utilizza un'architettura a livelli:

- TPM Device Driver (TDD): è il livello che contiene il driver per pilotare il TPM ed è fornito direttamente dal produttore del chip.

- TCG Device Driver Library (TDDL): è il livello che fornisce un'interfaccia standard detta TDDL Interface (TDDLI) verso il livello superiore. Esso permette l'accesso alle varie funzionalità fornite dal TPM qualsiasi sia il suo costruttore. Questo livello non supporta il multithreading quindi tutti i comandi per il TPM dovranno essere serializzati.

- TSS Core Services (TCS): è il livello che funziona come un servizio di sistema cioè fornisce le primitive e le funzioni che permettono di gestire le risorse del TPM. Esso permette di sincronizzare l'accesso al TPM fra le diverse applicazioni che si trovano al livello superiore. Espone un'interfaccia al livello superiore chiamata TCS Interface (TCSI);

- TCG Service Provider (TSP): è il livello che fornisce le funzionalità ausiliarie che è più comodo gestire al di fuori del TPM come la verifica delle firme o il calcolo dell'hash sui dati. Ogni applicazione si rivolgerà alla propria istanza TSP quindi in un sistema operativo con più processi in esecuzione ci potranno essere più istanze TSP. Questo livello espone un'interfaccia alle applicazioni chiamata TSP Interface (TSPI).

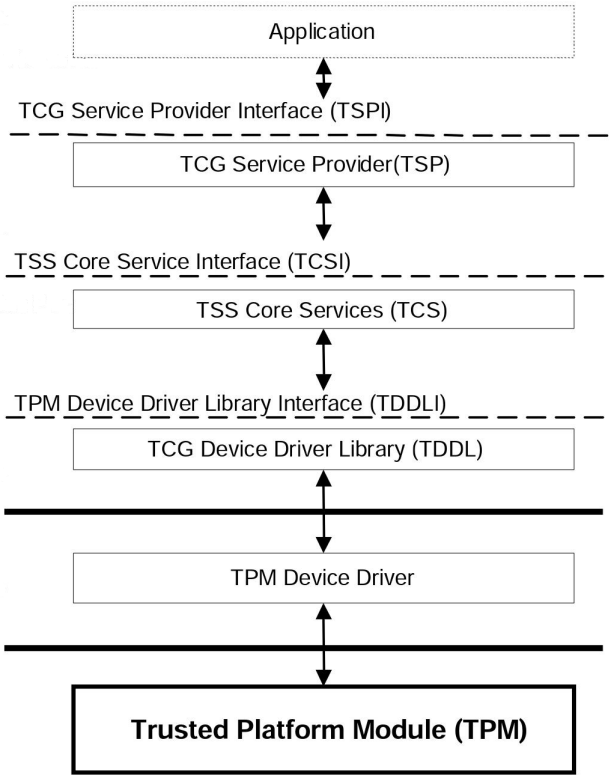

Figura 2.2. Architettura TSS (fonte:[6]).

2.3 Specifiche del TPM

L'attestazione remota è una delle funzionalità offerte dal TC. Il suo scopo è di dimostrare ad un'entità remota che il sistema operativo e le applicazioni software della propria piattaforma sono fidate. La Figura 2.3 illustra ad alto livello le entità coinvolte nel processo di attestazione ed i

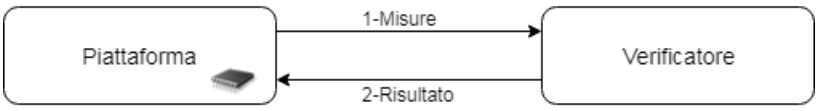

Figura 2.3. Attestazione remota ad alto livello.

dati scambiati. La piattaforma contiene il TPM attraverso cui verranno rilevate in maniera sicura le misure associate ai componenti del sistema ed in seguito verranno comunicate al verificatore. Il verificatore controlla i dati che ha ricevuto, li confronta con quelli di riferimento e comunica il risultato dell'attestazione alla piattaforma. Nelle sottosezioni successive verrà analizzato nel dettaglio l'evoluzione del meccanismo di attestazione remota dalla prima all'ultima versione TPM.

Il termine TPM indica sia il chip hardware sia le specifiche necessarie alla sua costruzione. Dopo la creazione del TCG si sono susseguite varie versioni che hanno migliorato alcune caratteristiche iniziali e che hanno permesso l'inserimento di nuove funzionalità. La prima versione fu pubblicata nel 2001 e venne indicata come TPM 1.1. Il problema principale di questa versione era la gestione della privacy nel processo di attestazione. L'aggiornamento noto come TPM 1.1b pose un rimedio temporaneo al problema della privacy utilizzando un meccanismo basato su una terza parte fidata. Nel 2003 si iniziò a lavorare alla nuova versione nota come TPM 1.2 che introdusse un nuovo meccanismo per eseguire l'attestazione. I primi attacchi all'algoritmo SHA-1 e la dimostrazione

della sua debolezza portarono il TCG allo sviluppo di una nuova versione che non obbligasse ad utilizzare uno specifico algoritmo ma permettesse una scelta più ampia, questa è l'ultima versione sviluppata e prende il nome di TPM 2.0.

Un sistema che deve eseguire l'attestazione deve essere in grado di fornire informazioni che riflettano lo stato attuale del sistema in esecuzione, informazioni globali cioè relative a tutto il sistema, limitare la divulgazione delle informazioni solo ai sistemi fidati, presentare il contenuto dell'attestazione in forma logica ed uniforme, utilizzare un meccanismo affidabile fra i sistemi coinvolti nella comunicazione [8].

2.3.1 TPM 1.1

Le entità coinvolte nel processo di attestazione sono la piattaforma, il Verifier e il privacy CA. La piattaforma è il sistema che contiene le applicazioni e il TPM. Il Verifier è il sistema che si occupa di confrontare le misure ricevute con quelle di riferimento. Il privacy CA è noto anche come Trusted Third Party (TTP) e garantisce la presenza di un TPM fidato e valido nella piattaforma.

La procedura di attestazione remota nel TPM 1.1 [9] si può dividere in due fasi.
La prima fase prevede l'interazione fra la piattaforma e il privacy CA, le operazioni svolte sono le seguenti:

1. il TPM genera una coppia di chiavi RSA dette Attestation Identity Key (AIK);
2. il TPM genera una struttura dati detta contenuto d'identità che principalmente sarà composta da:
   - AIK pubblica;
   - hash della chiave pubblica del privacy CA per dimostrare che la richiesta è rivolta proprio a quel privacy CA e contemporaneamente per evitare che la piattaforma richieda credenziali a più privacy CA.
3. il TPM genera una struttura dati detta identity proof che principalmente conterrà:
   - versione TPM utilizzata;
   - AIK pubblica;
   - firma del contenuto d'identità apposta con l'EK privata del TPM;
   - Endorsement Credential;
   - Platform Credential;
   - Conformance Credential.
4. il TPM genera una chiave di sessione che utilizza per cifrare simmetricamente l'identity proof;
5. il TPM crea la richiesta vera e propria da inviare al privacy CA inserendo:
   - identity proof cifrata;
   - chiave di sessione cifrata asimmetricamente con la chiave pubblica del privacy CA;
   - algoritmo simmetrico utilizzato per cifrare l'identity proof;
   - algoritmo asimmetrico utilizzato per cifrare la chiave di sessione.
6. il privacy CA riceve la richiesta ed estrae i dati, interpreta le credenziali che gli permettono di verificare la conformità della piattaforma rispetto alle specifiche e gli permettono di conoscere la chiave pubblica EK del TPM, verifica la firma del contenuto d'identità;
7. il privacy CA genera la credenziale per l'attestazione cioè un certificato contenente l'AIK pubblica e firmato utilizzando la propria chiave privata, questa credenziale afferma che l'AIK appartiene ad un TPM fidato ma non conterrà le informazioni specifiche del TPM per motivi di privacy;

8. il privacy CA genera una risposta contenente la credenziale per l'attestazione opportunamente cifrata simmetricamente con una chiave di sessione, tale chiave verrà comunicata sempre nella risposta ma sarà cifrata con la chiave EK pubblica del TPM;

9. il TPM riceve la risposta, decifra i dati verificando che siano corretti e memorizza la credenziale per l'attestazione.

La seconda fase prevede l'interazione fra la piattaforma e il Verifier, le operazioni svolte sono illustrate nella Figura 2.4:

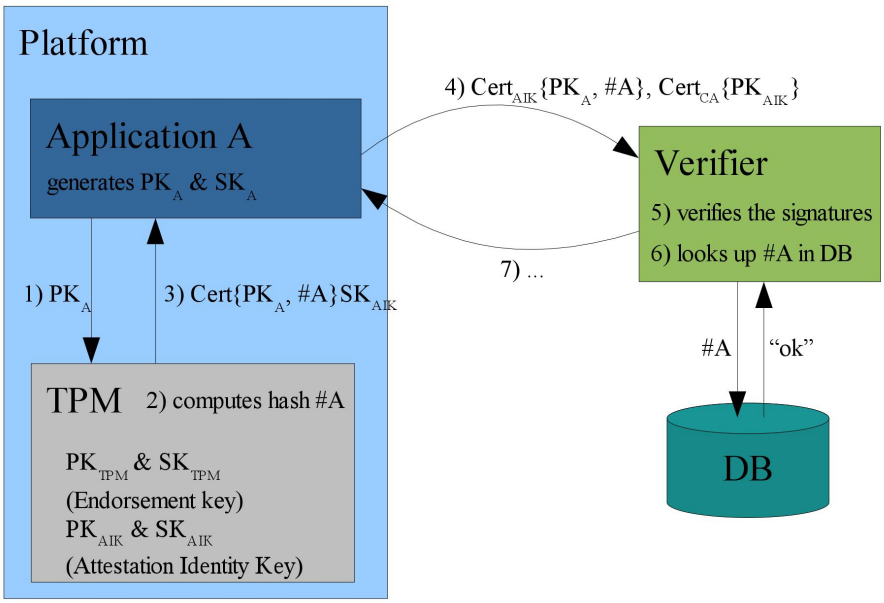

Figura 2.4. Interazione piattaforma-verificatore (fonte:[10]).

1. L'applicazione A genera asimmetricamente una coppia di chiavi pubblica (PK_A) e privata (SK_A) ed invia al TPM quella pubblica per certificarla;

2. il TPM calcola la misura cioè l'hash del codice eseguibile dell'applicazione #A;

3. il TPM crea un certificato contenente l'hash dell'applicazione e la chiave pubblica che potrà essere sfruttata in seguito per comunicare in maniera sicura con l'applicazione, firma questo certificato con la chiave privata AIK che aveva ottenuto nella prima fase di questo meccanismo;

4. l'applicazione vuole autenticarsi con un'entità remota ed invia il certificato ricevuto nella prima fase dal privacy CA con cui si specifica che la chiave pubblica AIK appartiene ad un TPM fidato ed il certificato contenente la chiave pubblica e l'hash dell'applicazione;

5. il Verifier controlla la catena di certificati che ha ricevuto;

6. il Verifier cerca l'hash #A nel proprio database e verifica il livello di fiducia assegnato a quell'applicazione;

7. il Verifier, se l'applicazione è affidabile, continua la comunicazione utilizzando la chiave pubblica PK_A per generare una connessione sicura.

Protocolli più sicuri prevedono anche la misurazione e l'invio dei dati relativi al sistema operativo, al firmware ed alle varie applicazioni presenti nel sistema.

Gli svantaggi dell'attestazione remota con privacy CA si devono cercare nell'eccessivo carico elaborativo del privacy CA che viene coinvolto in ogni transazione. Ogni volta che una piattaforma deve autenticarsi su un'entità remota dovrà richiedere il rilascio di un certificato con una nuova AIK perché se riutilizzasse sempre la stessa AIK sarebbe semplice per un'entità esterna tracciare il suo comportamento, il privacy CA diventa quindi un collo di bottiglia per l'attestazione remota. Un altro svantaggio è legato al livello di sicurezza con cui dovrebbe essere gestito il privacy CA. Esso conosce l'identità del TPM (tramite l'Endorsement Credential) e tutte le chiavi AIK che gli ha rilasciato quindi può tracciare il suo comportamento, siccome il privacy CA contiene informazioni critiche per la privacy dovrebbe essere implementato con un sistema molto robusto.

2.3.2 TPM 1.2

La versione 1.2 aggiunge numerose funzionalità come:

- la realizzazione di interfacce software standard;
- la protezione contro gli attacchi a dizionario cioè il tentativo di provare numerose password per capire quella che autorizza l'utente. Il TCG enuncia i principi con cui realizzare tale protezione ma l'effettiva implementazione è lasciata al produttore del TPM;
- le sessioni di trasporto che permettono una comunicazione sicura fra applicazioni e TPM tramite meccanismi che garantiscono confidenzialità ed integrità;
- l'assegnazione di una delega che permette di fornire temporaneamente il permesso di esecuzione di alcuni comandi ad una certa entità (persona o software). La revoca della delega implica l'impossibilità di continuare ad eseguire quei comandi. Nella versione precedente l'unico modo per permettere ad un'entità di utilizzare dei comandi era di fornirgli i dati di autenticazione ma questo permetteva a quell'entità di eseguire qualsiasi operazione e per poter bloccare i permessi si dovevano modificare i dati di autenticazione;
- la possibilità di tenere traccia dell'orario in cui una certa firma viene eseguita. Il TPM riesce a conteggiare i tick trascorsi dall'inizio della sessione e li converte nell'orario corretto grazie alla sincronizzazione con un orologio esterno. Non si può utilizzare un orologio interno al TPM perché le piattaforme su cui è inserito talvolta vengono spente quindi l'orologio non funzionerebbe correttamente;
- un nuovo metodo per l'attestazione che permette di sopperire ai problemi di privacy e prende il nome di Direct Anonymous Attestation (DAA);
- l'inserimento di nuove Application Programming Interface (API) mantenendo la compatibilità anche con le API della versione precedente.

Il meccanismo DAA [11] si basa su uno schema di firme di gruppo e sulla Zero Knowledge Proof (ZKP). Lo schema con firme di gruppo permette di inserire nei messaggi delle firme valide evitando che venga identificato l'utente che genera la firma. La ZKP permette di dimostrare la conoscenza di un segreto senza rivelarlo e si basa su un meccanismo a sfide. Il processo DAA può essere diviso in due fasi [7]:

1. Fase di join: è la fase in cui la piattaforma genera la propria richiesta di credenziali. L'emettitore verifica la richiesta ricevuta, controlla che la piattaforma contenga un TPM valido e se le verifiche vanno a buon fine emette le credenziali;
2. Fase di sign: è la fase in cui la piattaforma firma dei dati e dimostra di possedere le credenziali DAA ad un verificatore che controllerà la validità dei dati ricevuti e risponderà col risultato della verifica.

Un vantaggio rispetto al meccanismo di attestazione remota con privacy CA è l'utilizzo dell'emettitore solamente la prima volta che la piattaforma ha la necessità di ricevere le proprie

credenziali DAA. In seguito sarà la piattaforma stessa a generare le varie chiavi AIK che potrà utilizzare per firmare le misurazioni eseguite sul proprio software. Un altro vantaggio è la possibilità di revocare le credenziali ad una piattaforma senza avere la necessità di conoscere la sua identità. Gli svantaggi sono legati all'utilizzo di algoritmi crittografici che oggi sono ritenuti vulnerabili e le basse prestazioni legate all'esecuzione di questo meccanismo.

2.3.3 TPM 2.0

Nel 2014 venne pubblicata la nuova versione nota come TPM 2.0 [12], la principale novità fu la scelta di rendere il sistema interoperabile con vari algoritmi crittografici. Le strutture dati vennero modificate in modo da contenere al proprio interno i dati, l'algoritmo scelto e gli eventuali attributi necessari. Si diede, dunque, la possibilità di utilizzare sempre gli algoritmi più robusti e di supportare in seguito anche nuovi algoritmi. Per gli algoritmi asimmetrici venne aggiunto il supporto alle curve ellittiche con chiavi a 256 bit, per gli algoritmi simmetrici venne aggiunto il supporto all'AES 256, per gli algoritmi di hash ed HMAC venne aggiunto il supporto allo SHA-2 256.

Un'altra novità fu l'introduzione di un metodo unificato di autorizzazione. La versione precedente utilizzava metodi differenti a seconda che si dovesse autorizzare all'uso o alla delega o alla migrazione di oggetti mentre col TPM 2.0 c'è la possibilità di avere un meccanismo più flessibile con una granularità più fine relativamente all'accessibilità dei dati protetti e delle chiavi. Ogni comando che viene impartito al TPM necessita di un'adeguata autorizzazione. Il TPM 2.0 supporta tre meccanismi di autorizzazione:

1. password: viene trasmessa in chiaro ed è utilizzata solo nei casi in cui il canale attraverso cui avviene la comunicazione è sicuro;
2. HMAC: permette di dimostrare al TPM la conoscenza dei dati di autorizzazione senza comunicarli esplicitamente;
3. politiche: definiscono le condizioni necessarie per l'utilizzo di un oggetto. La politica potrebbe essere legata ad uno specifico valore del PCR oppure potrebbe limitare l'utilizzo di una chiave alla sola attestazione dei dati nei PCR.

L'accesso ad un oggetto non dipende solo dal meccanismo di autorizzazione utilizzato ma anche dal ruolo che quell'entità ricopre rispetto l'oggetto. Il TPM 2.0 supporta tre differenti ruoli di autorizzazione:

1. utente: ruolo necessario per svolgere le principali operazioni che utilizzano una chiave come l'inserimento di una firma o il caricamento di una chiave per decifrare dati che erano stati precedentemente memorizzati;
2. admin: ruolo necessario per svolgere operazioni avanzate come il controllo del certificato di un oggetto o il cambio del valore di autorizzazione associato all'oggetto;
3. dup: ruolo necessario per svolgere operazioni di duplicazione di oggetti.

Il controllo generale del TPM dipende da tre entità ognuna delle quali ha il proprio dominio di controllo:

1. Firmware della piattaforma: è realizzato dal produttore della piattaforma e gestisce l'hardware preparandolo ad essere utilizzato dal sistema operativo. Si occupa dell'allocazione della memoria non volatile del TPM, della configurazione dei PCR, del controllo delle gerarchie di chiavi, delle politiche e dei valori di autorizzazione;
2. Possessore della piattaforma: gestisce un sottoinsieme delle categorie di cui si occupa il firmware ed in particolare si occupa dell'allocazione di memoria non volatile e controlla la disponibilità delle gerarchie di chiavi per la memorizzazione;

3. Amministratore della privacy: gestisce la gerarchia di EK e segnala i dati sensibili per la privacy.

Spesso il possessore della piattaforma e l'amministratore della privacy vengono gestiti dalla stessa entità.

Il TPM 2.0 crea delle gerarchie ognuna delle quali ha le proprie password e politiche di autorizzazione, distinguiamo [13]:

- Platform Hierarchy (PH): viene controllata da un servizio della piattaforma come il BIOS o l'UEFI. Viene abilitata all'avvio della piattaforma e permette di controllare l'integrità del firmware, in seguito dà la possibilità di inizializzare il TPM e di abilitare le altre gerarchie;
- Storage Hierarchy (SH): è controllata dal possessore del TPM quindi i dati associati ad essa vengono cancellati quando cambia il possessore. Permette di proteggere chiavi e password che non sono sensibili alla privacy;
- Endorsement Hierarchy (EH): è controllata dall'amministratore della privacy. I dati legati all'utente vengono cancellati quando si cambia il possessore mentre i dati legati al produttore del dispositivo saranno validi per l'intero ciclo di vita della piattaforma. Serve a proteggere chiavi e password che sono sensibili alla privacy;
- Null Hierarchy (NH): è utilizzata quando il TPM deve svolgere il compito di coprocessore crittografico. I dati presenti vengono cancellati ogni volta che la piattaforma viene spenta.

Ogni dominio del TPM è legato ad una di queste gerarchie. Il dominio di sicurezza in cui le funzioni proteggono la sicurezza dell'utente è legato alla SH. Il dominio della privacy in cui le funzioni proteggono l'identità della piattaforma e dell'utente è legato alla EH. Il dominio della piattaforma in cui le funzioni proteggono l'integrità dei servizi della piattaforma e del firmware è legato alla PH. Le gerarchie sono entità persistenti che non possono essere cancellate ma solo abilitate e disabilitate mentre possono essere cancellati i dati associati ad esse. Nel TPM 2.0 il controllo delle funzionalità di firma, attestazione e cifratura dipende dalle gerarchie EH ed SH, mentre le funzionalità di gestione dipendono dalla PH, ogni gerarchia ha il proprio possessore. Nel TPM 1.2 c'era solo un singolo possessore che eseguiva la firma di attestazione con una EK (chiave RSA 2048) ed eseguiva la cifratura simmetrica con una SRK (chiave RSA 2048) cioè un singolo utente aveva il controllo su tutte le operazioni di firma, attestazione e cifratura.

Un'altra differenza rispetto il TPM 1.2 è l'utilizzo di poche tipologie di chiavi. Gli attributi associati ad una chiave potranno essere:

- restricted: restringe il campo di azione di una chiave. Non è possibile generare una chiave che abbia solo questo attributo attivo;
- sign: specifica la possibilità di firmare dei dati;
- decrypt: specifica la possibilità di decifrare dei dati.

Una chiave potrà essere caratterizzata da una combinazione dei precedenti attributi. Ciò permette di mantenere la compatibilità con le altre tipologie di chiavi utilizzate nel TPM 1.2. Ad esempio nel TPM 1.2 una chiave d'identità permette di firmare solo strutture prodotte all'interno del TPM, si può ottenere lo stesso comportamento nel TPM 2.0 assegnando gli attributi sign e restricted alla chiave.

Il TPM 2.0 introduce i ticket che sono una struttura dati contenente un HMAC, essi permettono di validare delle operazioni eseguite dal TPM in modo che successivamente non sarà necessario rieseguirle. Vengono utilizzati per rendere più veloci delle operazioni ad esempio dopo che il TPM verifica una firma asimmetrica può decidere di salvare quel digest utilizzando HMAC ed una propria chiave segreta così la successiva verifica sarà più veloce. La chiave segreta utilizzata è nota solo al TPM e ciò dà autenticità, inoltre il meccanismo HMAC fornisce integrità dei dati. Distinguiamo 5 tipi di ticket:

1. Ticket di creazione: è creato insieme alla generazione di un nuovo oggetto ed è usato successivamente per certificare che l'oggetto e tutti i parametri ad esso collegati siano stati creati correttamente;

2. Ticket di verifica: è creato quando si verifica una firma e permette di ottenere l'autorizzazione cifrata utilizzando la crittografia simmetrica, ciò permette di risparmiare tempo e di migliorare le prestazioni per le verifiche successive;

3. Ticket di autorizzazione: è collegato all'autorizzazione delle politiche e permette di verificare i parametri di autorizzazione e per quanto tempo sarà valida quell'autorizzazione;

4. Ticket di hash: generato su dei dati provenienti dall'esterno del TPM, si usa per indicare che i dati su cui è stato generato il digest sono stati controllati dal TPM e sono sicuri quindi su di essi può essere eseguita la firma;

5. Ticket NULL: è generato quando ci si attenderebbe un ticket come risposta ma in realtà non viene prodotto nessun ticket; un esempio di utilizzo è quando si dovrebbe produrre un ticket di autorizzazione per un oggetto con un tempo di validità quasi nullo ed in questo caso si preferisce fornire direttamente questo tipo di ticket.

Un miglioramento rispetto la versione TPM 1.2 è la protezione da attacchi a dizionario. Il TPM 2.0 introduce un meccanismo standard di protezione. Quando vengono rilevati dei tentativi di inserimento consecutivo di chiavi di autorizzazione che risultano sempre errate si entrerà nella modalità di lockout. Il TPM si proteggerà impedendo l'autorizzazione per un certo tempo a meno che non si inserisca un altro valore segreto immagazzinato nel registro lockoutAuth.

L'attestazione è il meccanismo che permette al TPM di firmare dei suoi dati interni e può avvenire per differenti tipi di dati come i dati nei PCR, i dati dell'orologio e i dati legati ad altri oggetti TPM. Per eseguire l'attestazione il TPM genera una struttura standard che conterrà anche informazioni sull'orario e sul firmware del TPM ed aggiunge i dati specifici per l'attestazione, poi viene calcolato l'hash del blocco di dati risultanti e viene firmato con la chiave di firma selezionata. La combinazione della versione del firmware più le informazioni sul clock potrebbero essere usate per collegare due attestazioni differenti provenienti dallo stesso TPM e per prevenire tale comportamento si esegue l'offuscamento di tali valori. La principale differenza rispetto al DAA del TPM 1.2 è l'utilizzo di algoritmi basati su curve ellittiche che hanno il vantaggio, a parità di sicurezza, di essere più veloci e di avere chiavi più piccole; in questo caso il meccanismo prenderà il nome di ECDAA. La struttura del meccanismo di attestazione resta uguale a quella presente nel TPM 1.2, gli elementi coinvolti nell'attestazione ECDAA sono l'emettitore, la piattaforma (costituita dal TPM e dalla restante parte del sistema che prende il nome di host) e il Verifier. Le varie fasi del meccanismo ECDAA sono [14]:

1. Setup: è la fase iniziale in cui ogni entità genera i parametri necessari al proprio funzionamento. L'emettitore genera la propria coppia di chiavi asimmetriche e stabilisce i parametri necessari alla costruzione e gestione della curva ellittica. Il TPM associato alla piattaforma genera la propria coppia di chiavi asimmetriche ed un valore casuale che fungerà da segreto. Alla fine di questa fase vengono resi pubblici i parametri della curva ellittica e le chiavi pubbliche per evitare che debbano essere inviati in input nelle successive fasi;

2. Join: è la fase in cui la piattaforma dimostra all'emettitore di possedere un TPM fidato ed ottiene le credenziali di gruppo, l'algoritmo utilizzato assume che piattaforma ed emettitore comunichino su un canale sicuro;

3. Sign: è eseguita all'interno della piattaforma con la comunicazione fra il TPM e la parte host e serve a produrre una firma basata sulla credenziale ricevuta nella fase precedente dall'emettitore e sul segreto che aveva generato il TPM;

4. Verify: è la fase eseguita nel Verifier per controllare la validità di una firma;

5. Link: è una fase opzionale eseguita dal Verifier per controllare se due firme valide sono collegate fra loro.

I comandi TPM 2.0 coinvolti durante le varie fasi dell'attestazione sono:

- TPM2_Create: è usato per generare una coppia di chiavi asimmetriche, prende in input dei parametri pubblici come ad esempio le informazioni sulla curva ellittica che si dovrà utilizzare;
- TPM2_Load: permette di caricare la coppia di chiavi asimmetriche nel TPM. Dopo che le chiavi sono state create verranno opportunamente cifrate ed immagazzinate all'esterno del TPM. Con questo comando è possibile acquisire il blob di dati contenenti le chiavi che verrà decifrato e su di esso si eseguirà una verifica di integrità, se la verifica va a buon fine le chiavi verranno caricate all'interno del TPM;
- TPM2_ActivateCredential: permette di attivare le credenziali cioè viene utilizzato quando il TPM riceve una sfida dall'emettitore, tramite la decifratura e la restituzione del numero casuale dimostra di possedere la chiave privata necessaria a rispondere correttamente;
- TPM2_Commit: viene utilizzato nella fase di sign e permette di calcolare la prima parte della firma per cui vengono utilizzati i parametri associati alla curva ellittica;
- TPM2_Sign: permette di calcolare la seconda parte della firma acquisendo il digest del messaggio che si vuole firmare e calcolando su di esso la firma.

Capitolo 3

Reti virtuali

In questo capitolo introdurremo il concetto di virtualizzazione nella Sezione 3.1. A seguire tratteremo le caratteristiche del paradigma Network Function Vitualisation (NFV) nella Sezione 3.2 ed infine analizzeremo l'architettura NFV e le sue funzionalità nella Sezione 3.3.

3.1 La virtualizzazione

Il termine virtualizzazione indica tecniche che permettono l'astrazione dai componenti fisici sottostanti in modo da renderli disponibili come risorse virtuali. Tramite appositi software le risorse virtuali (processore, RAM, disco fisso, scheda di rete) vengono gestite come se fossero presenti in un sistema reale e ciò dà la possibilità di creare le cosiddette macchine virtuali. Ogni macchina virtuale avrà il proprio hardware virtuale ed in essa potranno venire installati un sistema operativo e delle applicazioni. La gestione delle risorse virtuali [15] è caratterizzata da:

- isolamento: le risorse non hanno alcuna dipendenza dal sistema fisico;
- emulazione: la possibilità di offrire risorse differenti da quelle presenti nel sistema fisico;
- condivisione: l'associazione di più risorse virtuali alla stessa risorsa fisica;
- aggregazione: l'unione di più risorse fisiche in un'unica risorsa virtuale.

La virtualizzazione è possibile grazie ad uno strato software detto hypervisor, il cui scopo è l'allocazione e la gestione delle risorse per permettere l'esecuzione delle macchine virtuali. Un hypervisor viene detto di tipo 1 quando è direttamente integrato nell'hardware del sistema ospitante (VMWare ESXi [16], Microsoft Hyper-V [17], Xen [18]), viene detto di tipo 2 se viene eseguito come una normale applicazione nel sistema ospitante (KVM [19], VMWare Player [20], Oracle VirtualBox [21]). La Figura 3.1 illustra i due tipi di hypervisor.

Il concetto di virtualizzazione nasce già negli anni '60 [22] quando si presentò la necessità di dividere logicamente le risorse disponibili in un mainframe fra le diverse applicazioni presenti. I mainframe erano dei sistemi molto costosi e il loro funzionamento originale prevedeva la possibilità di eseguire un programma alla volta dunque spesso il mainframe era sottoutilizzato e ciò non giustificava un investimento così elevato da parte delle aziende. IBM, la maggior azienda produttrice di mainframe, diede una spinta alla loro diffusione grazie allo sviluppo del primo sistema commerciale (CP-67) che permettesse la creazione di un ambiente con macchine virtuali. Questo approccio era nettamente diverso da quello utilizzato finora basato sull'utilizzo del sistema a condivisione di tempo cioè ogni utente metteva in coda il proprio processo che veniva eseguito al suo turno (batch processing). La virtualizzazione permise ad ogni utente di avere il proprio sistema operativo nella macchina virtuale e di avere la possibilità di utilizzare le risorse disponibili nel mainframe inoltre migliorò la sicurezza e l'affidabilità perché ogni macchina virtuale era separata dalle altre.

Negli anni '90 [23], con l'affermazione di elaboratori basati sull'architettura x86 sia in ambiente server sia desktop, si ripresentarono alcuni problemi già affrontati per i mainframe come:

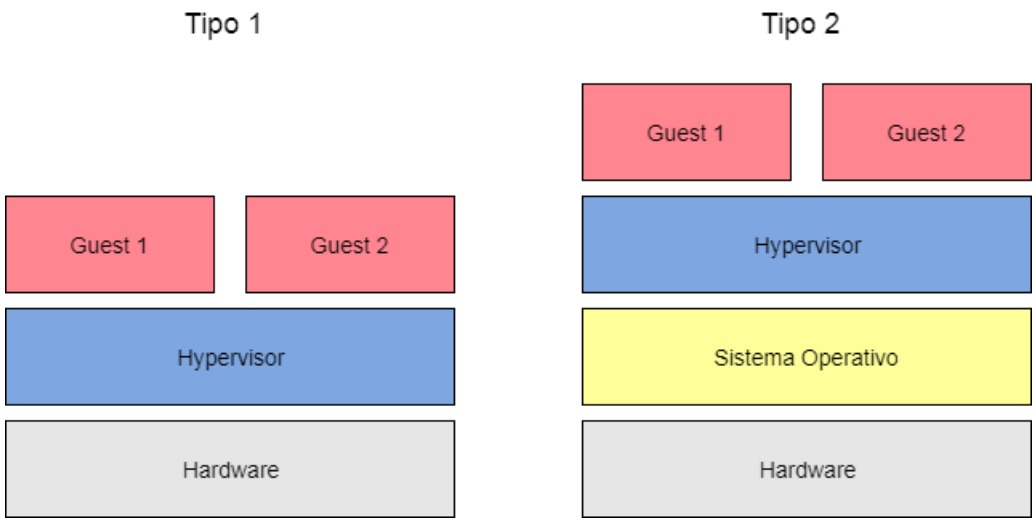

Figura 3.1. Hypervisor tipo 1 e tipo 2.

- sottoutilizzo: generalmente i server venivano utilizzati solo per una piccola percentuale rispetto alla loro capacità complessiva e per evitare problemi legati alla sicurezza si tendeva ad utilizzare solo un'applicazione per server;
- aumento dei costi: i server dovevano rimanere sempre operativi quindi i costi legati al loro consumo e al loro raffreddamento non erano proporzionali al loro effettivo utilizzo; anche il costo per formare il personale incaricato a gestire i sistemi informatici aumentava con l'aumentare della complessità del sistema;
- inaccessibilità temporanee: server che fornivano servizi critici dovevano permettere un accesso continuo a tali servizi quindi bisognava proteggersi da attacchi informatici ed anche da eventuali fenomeni naturali che avrebbero potuto danneggiare il server.

Qualche anno dopo, VMware introdusse i virtual desktop che permisero la creazione di macchine virtuali nei computer. I principali vantaggi della virtualizzazione sono:

- ottimizzazione delle risorse: si sfruttano le risorse dei server in maniera significativa facendo girare varie applicazioni sul server grazie alla separazione ottenuta attraverso le macchine virtuali;
- riduzione dei costi: si riduce il numero di server necessari per fornire i propri servizi con la conseguente riduzione dei costi legati allo spazio, all'alimentazione e al raffreddamento;
- maggiore flessibilità operativa: gli amministratori di sistema possono dedicare un tempo minore ad attività come la configurazione, il monitoraggio e la gestione della manutenzione dei sistemi informatici aziendali;
- eliminazione tempi di inattività: è possibile fornire un servizio continuo pianificando delle migrazioni verso altri ambienti.

3.2 Network Function Virtualisation

I principi di virtualizzazione discussi nella Sezione 3.1 possono essere applicati anche alle funzionalità offerte dalle reti. I problemi principali con cui si devono confrontare i fornitori di servizi di rete sono legati alla grande varietà di hardware proprietario che è necessario installare o aggiornare per fornire un nuovo servizio. L'impatto economico dovuto allo spazio fisico su cui aggiungere i nuovi dispositivi, l'incremento nel consumo di energia, la formazione di personale specializzato

nella gestione degli apparati hardware per il nuovo servizio sono problemi che potrebbero essere risolti con la virtualizzazione delle funzioni di rete.

Nel 2012 [24] lo European Telecommunications Standards Institute (ETSI) creò un gruppo di lavoro (Industry Specification Group - ISG) col compito di definire i requisiti e l'architettura per permettere la virtualizzazione delle funzioni di rete indicate come Network Function Virtualisation (NFV). L'ISG è formato da società operanti nel settore delle reti, da fornitori di servizi informatici e da venditori di prodotti di telecomunicazioni e le sfide che si trova ad affrontare sono:

- assicurarsi che le piattaforme con reti virtualizzate siano più semplici di quelle esistenti;
- creare applicazioni che funzionino con piattaforme hardware ed hypervisor differenti;
- mantenere delle prestazioni accettabili usando hypervisor e tecnologie software adeguati;
- gestire ed orchestrare delle applicazioni nelle reti virtuali;
- sicurezza contro attacchi o cattive configurazioni;
- stabilità della rete durante il caricamento o lo spostamento delle applicazioni;
- capacità di fornire il servizio anche in presenza di guasti dell'hardware o del software;
- minimizzare il consumo di energia.

NFV nacque dall'idea di disaccoppiare il software dall'hardware di un dispositivo di rete. Si capì che l'hardware specifico utilizzato fino a quel momento per realizzare le funzionalità di rete poteva essere rimpiazzato da un dispositivo hardware generico detto Commercial Off The Shelf (COTS) nel quale venisse implementato del software che fornisse la funzionalità di rete desiderata. È necessario un ulteriore strato intermedio fra hardware e software che è l'hypervisor il quale permette l'astrazione dall'hardware sottostante e la realizzazione della macchina virtuale. La Figura 3.2 mostra il differente approccio fra la modalità classica in cui ogni funzione di rete ha il proprio hardware specifico su cui viene implementata e il nuovo approccio NFV in cui si utilizza hardware COTS ed un orchestratore per implementare dei servizi di rete virtuali utilizzando l'astrazione dell'hardware per la computazione, la memorizzazione e la connettività. L'orchestratore ha il compito di gestire sia le risorse fisiche sia le funzioni virtualizzate. I servizi di rete possono essere istanziati o mossi in varie posizioni nella rete senza la necessità di installare nuovi dispositivi. Una delle caratteristiche più importanti introdotta dalle reti NFV è la scalabilità cioè la possibilità di aumentare o diminuire le risorse assegnate ad una VM dinamicamente in base al carico attuale del sistema. Nel caso in cui siano necessarie maggiori risorse computazionali sarà possibile assegnare alla VM più risorse fisiche del server nella quale essa si trova e questa operazione è chiamata scale up, l'operazione opposta che permette di liberare risorse precedentemente assegnate ad una VM è detta di scale down. Inoltre è anche possibile duplicare la VM su più server fisici ed utilizzarla logicamente come se fosse un unico sistema costituito dalle risorse complessive dei server, in questo caso ci dovrà essere anche un bilanciatore del carico per il traffico in arrivo e questa operazione è chiamata scale out mentre l'operazione duale viene detta di scale in.

I principali vantaggi del NFV sono:

- riduzione del consumo di energia e del numero di dispositivi necessari per supportare il servizio da sviluppare;
- minor tempo necessario per sviluppare nuovi servizi di rete;
- possibilità di usare una singola piattaforma per implementare più servizi;
- maggiore flessibilità grazie alle operazioni di scalabilità;
- opportunità di provare nuovi servizi con un basso fattore di rischio;
- evoluzione indipendente dell'hardware e del software di un dispositivo.

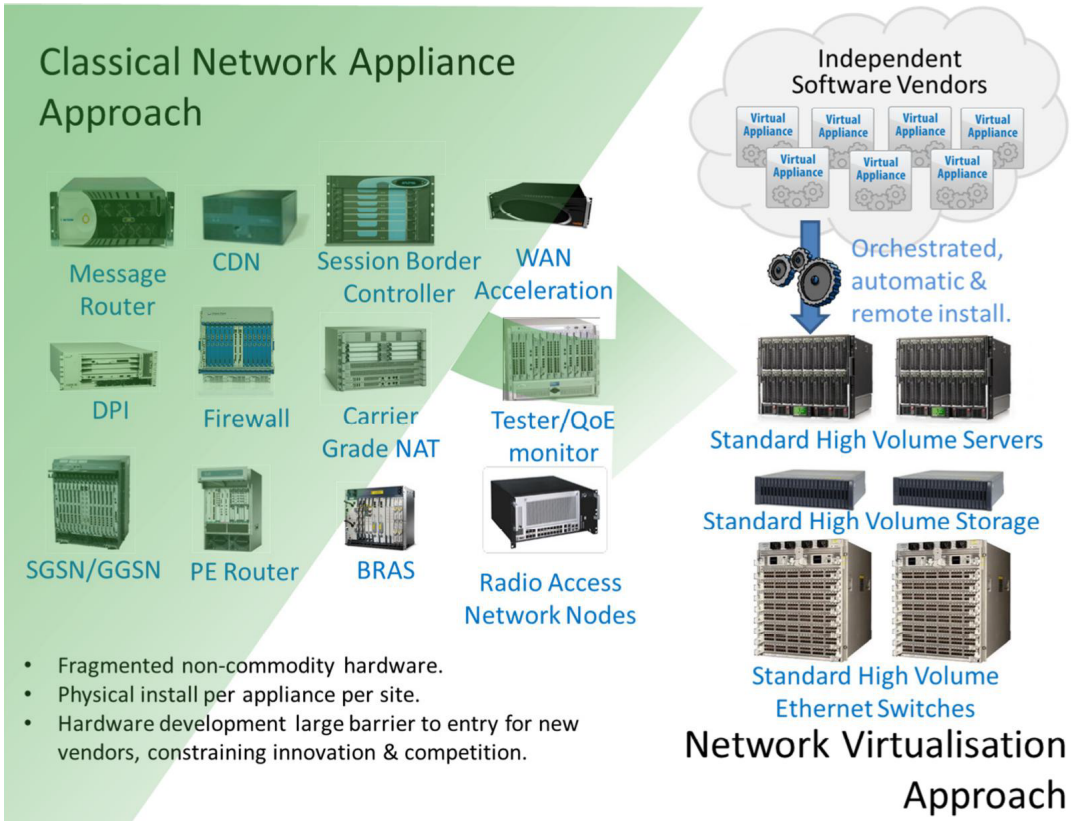

Figura 3.2. Approccio Classico ed approccio NFV (fonte:[24]).

Parallelamente ed in maniera indipendente da NFV si sviluppa un nuovo approccio detto Software Defined Networking (SDN) [25] che permette di programmare la rete. Esso è caratterizzato dal disaccoppiamento fra piano dati e piano di controllo. I nodi di rete diventano più semplici perché contengono solo l'hardware per inoltrare i pacchetti. Sarà necessario un sistema operativo di rete che riceve i comandi dal controllore e li inserisce nei nodi. La modalità di funzionamento è reattiva cioè quando un nodo riceve un nuovo pacchetto lo invia al controllore che decide il percorso da fargli seguire ed aggiorna le tabelle interne di ogni nodo con le nuove regole di inoltro. SDN permette di centralizzare la gestione del piano di controllo in cui si eseguono le scelte sul percorso da far seguire al traffico mentre lascia distribuito nei vari nodi il piano dati che permette di inoltrare i pacchetti verso la destinazione scelta. L'interazione fra i componenti SDN avviene tramite interfacce ben definite, in particolare l'interfaccia fra le applicazioni e il sistema operativo di rete viene detta di northbound mentre quella fra sistema operativo di rete ed hardware viene detta di southbound. Il compito del sistema operativo di rete sarà di fornire una visione astratta della rete alle applicazioni attraverso l'interfaccia di northbound e di programmare il comportamento dei singoli nodi di rete attraverso l'interfaccia di southbound, ciò è ben diverso dall'approccio tradizionale in cui è necessario configurare ogni singolo dispositivo della rete.

I principali vantaggi del SDN sono:

- una migliore gestione delle risorse di rete presenti. Si possono implementare meccanismi per smistare il traffico lungo i percorsi meno carichi semplicemente realizzandoli all'interno del controllore oppure si possono isolare delle zone della rete al fine di eseguire dei test;
- una maggiore scalabilità ed affidabilità grazie alla centralizzazione del controllo;
- la possibilità di utilizzare prodotti di differenti aziende che non conterranno al proprio interno funzioni di gestione liberando così i fornitori di servizi dalla necessità di configurare e supportare numerosi e differenti protocolli.

La comunicazione tramite l'interfaccia di southbound avviene attraverso un protocollo chiamato Openflow [26] che è diventato lo standard de facto. Openflow permette il mappaggio delle regole stabilite dal controllore all'interno dei vari nodi. Il suo funzionamento prevede di controllare uno dei campi del pacchetto in ingresso come l'indirizzo MAC o l'IP grazie al quale individua la regola da applicare che potrebbe essere l'inoltro su una certa porta o l'invio al controllore o l'eliminazione del pacchetto. Anche se tipicamente SDN ed Openflow utilizzano l'approccio reattivo, è possibile utilizzare anche un'altra modalità operativa che è quella proattiva la quale prevede che le regole di inoltro vengano inserite nei vai nodi prima di iniziare a ricevere del traffico in modo da sapere fin da subito come trattare i vari pacchetti che arrivano.

I paradigmi NFV ed SDN nascono e possono essere utilizzati in maniera indipendente ma in realtà la loro combinazione permette di ottenere i migliori risultati perché si uniscono i vantaggi della centralizzazione e semplicità di implementazione alla possibilità di astrarsi dall'hardware per la realizzazione di funzionalità di rete. Un esempio tipico del loro utilizzo congiunto è la creazione di catene di servizi realizzate con funzioni virtuali tramite la creazione di un apposito percorso che attraversi le VM opportune ognuna delle quali può implementare un servizio differente. È possibile offrire servizi differenti ad utenti differenti semplicemente distinguendo il loro traffico cioè creando delle catene che attraversano alcune o tutte le VM presenti. La modalità utilizzata sarà quella proattiva infatti ogni pacchetto in ingresso seguirà il percorso prestabilito attraverso i server e le relative VM. I vantaggi dell'utilizzo di SDN ed NFV risiedono nella velocità con cui è possibile fornire nuove catene di servizi grazie all'instradamento gestito via software da un controllore, nella gestione dei dispositivi fisici che non dovranno essere cambiati ogni volta che si deve fornire un nuovo servizio e nella possibilità di creare catene differenti per utenti differenti.

3.3 Architettura NFV

Il paradigma NFV disaccoppia il software dall'hardware quindi anche l'architettura necessaria alla sua realizzazione rispecchia questo cambiamento [27] introducendo un ulteriore elemento che permetta di sincronizzare il software con l'hardware sottostante e gestisca gli eventuali guasti. La Figura 3.3 illustra ad alto livello le tre macro entità di cui è costituita l'architettura NFV:

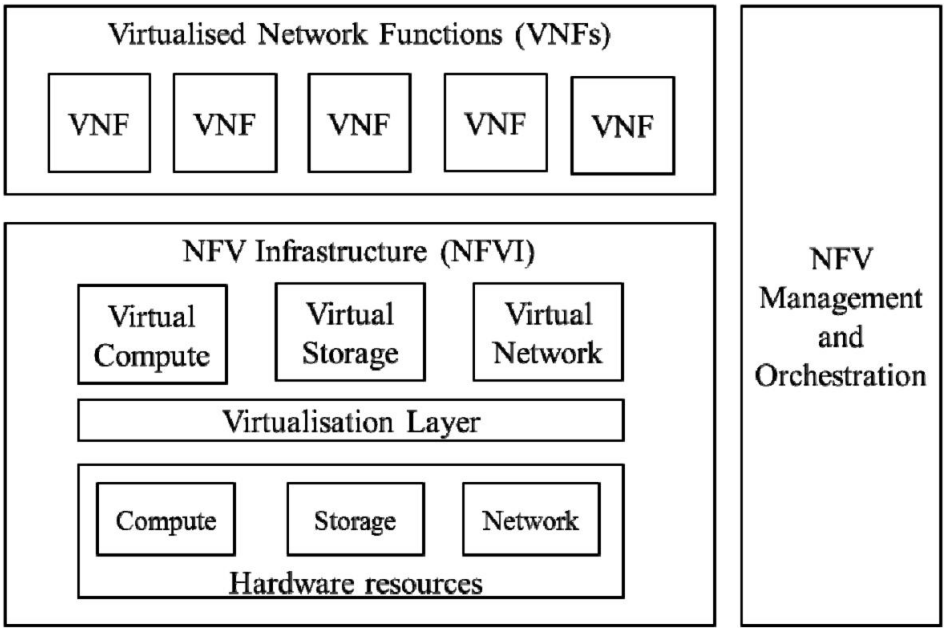

Figura 3.3. Visione ad alto livello dell'architetura NFV (fonte:[27]).

1. NFV Infrastructure (NFVI): include le risorse fisiche che permettono di implementare la computazione, la memorizzazione e le reti, inoltre contiene un livello di virtualizzazione

che permette di astrarsi dall'hardware sottostante ed espone le corrispondenti funzionalità virtuali;

2. Virtualised Network Function (VNF): è l'implementazione di una funzione di rete eseguita sopra NFVI;

3. NFV Management and Orchestration (NFV MANO): nasce dalla necessità di disaccoppiare le VNF dalle risorse hardware. I compiti che gestisce sono il mappaggio dei servizi di rete nei corrispondenti elementi NFV, l'istanziazione delle VNF nelle locazioni appropriate, l'allocazione e la scalabilità delle risorse, il tracciamento delle istanze VNF, il rilevamento dei guasti e la loro correzione.

NFV permette di implementare dei servizi di rete (Network Service - NS). Ogni NS può essere visto come la composizione di più funzioni di rete (Network Function - NF). Ogni NF ha delle interfacce ed un comportamento ben definiti e nel caso in cui sia realizzata attraverso un'infrastruttura virtualizzata prenderà il nome di VNF. Il flusso di attraversamento delle reti virtuali è una caratteristica importante per la creazione di catene di servizi ed essa verrà gestita da un'apposita entità che è il VNF Forwarding Graph (VNFFG).

La Figura 3.4 mostra l'architettura NFV completa di tutti i blocchi funzionali che intervengono

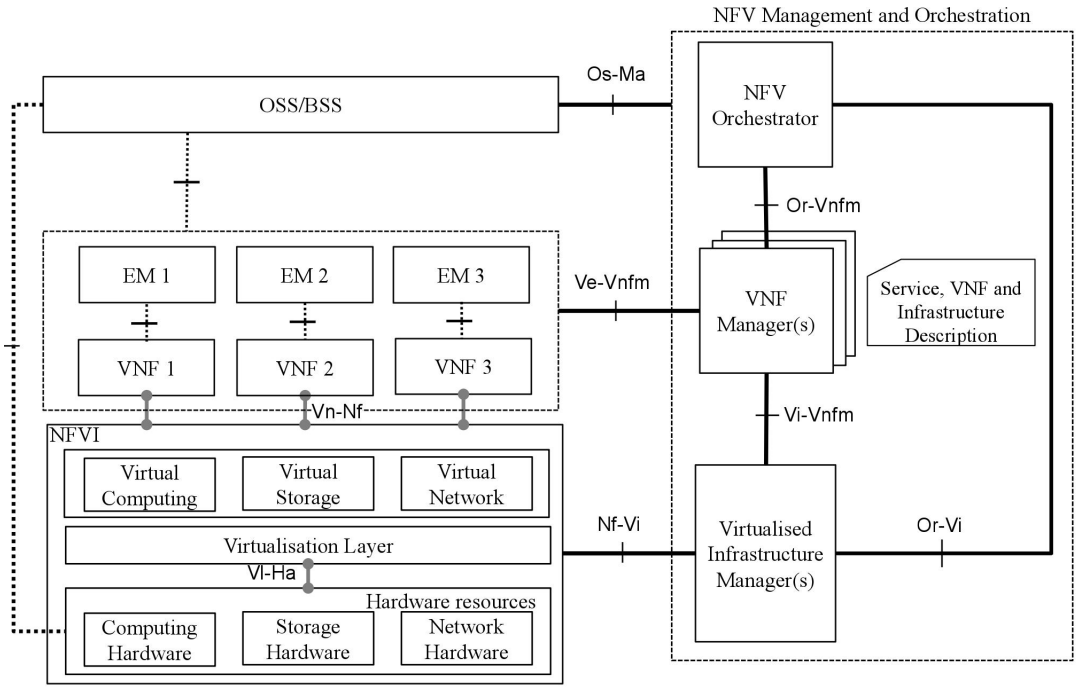

Figura 3.4. Visione completa dell'architetura NFV (fonte:[27]).

durante lo sviluppo dei servizi di rete ed esplicita le interfacce di comunicazione fra le varie entità:

- VNF: è la virtualizzazione di una funzione di rete e può essere sviluppata su una o più VM, le VM saranno collegate fra loro internamente e nel complesso permetteranno di realizzare la funzione desiderata; la VNF deve avere lo stesso comportamento dell'equivalente physical network function (PNF) cioè della corrispondente funzione implementata direttamente sull'hardware senza livelli di virtualizzazione;

- Element Management (EM): permette di implementare le funzionalità di gestione di uno o più VNF. Svolge le operazioni di configurazione delle VNF quindi ci potrà essere un unico EM per più VNF se deve essere caricata la stessa configurazione su tutte le VNF oppure ci potrà essere un EM che gestisca la configurazione di una singola VNF;

- Risorse hardware: fanno parte del NFVI e sono l'insieme di risorse che permettono di fornire computazione, memorizzazione e connettività alle VNF. Tipicamente l'hardware è COTS e le funzionalità di computazione e memorizzazione si trovano sullo stesso dispositivo mentre le funzioni di rete sono implementate con dei dispositivi di rete e i relativi collegamenti cablati o senza fili;
- Livello di virtualizzazione: fa parte del NFVI e permette di astrarsi dalle risorse hardware consentendo un'evoluzione indipendente dell'hardware dalle VNF. Esso si occupa di partizionare logicamente le risorse fisiche e di fornire alle VNF le risorse così virtualizzate, tipicamente questa funzionalità è eseguita da un hypervisor. La virtualizzazione delle reti permette di creare collegamenti fra le VM di una VNF o fra più VNF e si ottiene con tecniche come le virtual local area network (VLAN) o le virtual extensible local area network (VxLAN);
- Virtualised Infrastructure Manager (VIM): fa parte del NFV MANO ed è l'entità che controlla e gestisce l'infrastruttura. Si occupa dell'allocazione e del rilascio delle risorse, dell'analisi delle prestazioni, della collezione di informazioni relative ai guasti, della collezione di informazioni per la pianificazione, il monitoraggio e l'ottimizzazione;
- NFV Orchestrator (NFVO): fa parte del NFV MANO ed è l'entità che si occupa di gestire e far funzionare l'architettura NFV nel suo complesso quindi sia l'infrastruttura sia le risorse software. Questo componente gestisce i template di sviluppo dei NS eseguendo una procedura di validazione prima di caricarli e di provare ad istanziare la VNF, valida ed autorizza le richieste di risorse fatte dal VNFM, gestisce la creazione, l'aggiornamento e la terminazione dell'istanza del servizio di rete;
- VNF Manager (VNFM): fa parte del NFV MANO ed è l'entità che si occupa di gestire il ciclo di vita delle VNF quindi si dedica alla loro istanziazione, all'aggiornamento, alla scalabilità e alla terminazione;
- Descrittori: contengono le informazioni relative allo sviluppo delle VNF, all'inoltro dei dati attraverso le VNF e all'infrastruttura;
- Operations support system/Business support system (OSS/BSS): è un'interfaccia verso il fornitore di servizi di rete.

Noto il funzionamento di ogni componente è possibile capire il tipo di informazioni scambiate attraverso le interfacce di comunicazione fra i vari componenti. Ci saranno interfacce interne che permettono la comunicazione fra elementi interni ad una delle tre macro entità ed interfacce esterne che mettono direttamente in comunicazione le macro entità:

- L'interfaccia fra VNF ed NFVI è legata alla virtualizzazione delle risorse in modo che le VNF possano essere spostate da un NFVI ad un altro (portabilità) o che si possa utilizzare hardware differente per sviluppare le stesse VNF (indipendenza);
- L'interfaccia fra l'hardware ed il livello di virtualizzazione è interna al NFVI e permette di creare l'ambiente di esecuzione per le VNF e di collezionare le informazioni sullo stato delle risorse hardware;
- L'interfaccia fra orchestratore e VNFM è interna ad NFV MANO e permette di eseguire richieste collegate alle risorse, di scambiare informazioni sulla configurazione delle VNF e di collezionare informazioni utili per gestire il ciclo di vita del servizio di rete;
- L'interfaccia fra VIM e VNFM è interna al NFV MANO ed è usata per richiedere l'allocazione delle risorse, per configurare le risorse hardware e per scambiare le informazioni di stato;
- L'interfaccia fra orchestratore e VIM è interna ad NFV MANO ed è usata per richieste di allocazione, per riservare delle risorse, per configurare le risorse hardware e per scambiare le informazioni di stato;
- L'interfaccia fra NFVI e VIM è usata per richiedere ed assegnare le risorse virtualizzate, per scambiare le informazioni di stato sulle risorse virtualizzate e sull'hardware;

- L'interfaccia fra VNF e VNFM è usata per gestire il ciclo di vita delle VNF, per scambiare informazioni di configurazione ed informazioni sullo stato delle VNF;
- L'interfaccia fra OSS/BSS e NFV MANO è usata per gestire il ciclo di vita del servizio di rete e delle VNF di cui è composto, per scambiare informazioni sullo stato dei componenti, per gestire le politiche, per scambiare i dati da analizzare, per informazioni sulle risorse NFVI.

3.3.1 Funzionalità NFV

L'obiettivo che si è proposto l'ISG nella realizzazione dell'architettura NFV è stato di creare un'architettura in cui i vari elementi fossero indipendenti l'uno dall'altro in modo da non avere vincoli di nessun tipo ma si è scontrato anche con la necessità di creare dei meccanismi che avessero almeno le stesse caratteristiche degli equivalenti fisici [28].

L'architettura NFV permette agli operatori di rete di virtualizzare le funzioni di rete necessarie a creare, sviluppare e gestire il servizio; la virtualizzazione potrà essere parziale se alcune funzioni vengono ancora realizzate in hardware o totale se tutto viene realizzato in software. In entrambi i casi NFV permette lo sviluppo del servizio anche se gli ambienti fisici sono gestiti da operatori differenti.

La capacità di caricare, eseguire e spostare le VNF su ambienti differenti appartenenti anche a fornitori di servizi differenti viene detta portabilità, NFV ottimizza la localizzazione e l'allocazione delle risorse necessarie a spostare la VNF fra i differenti ambienti. Questa caratteristica permette di offrire anche un servizio continuo infatti in caso di guasto NFV è in grado di fornire meccanismi che ripristinino le istanze VNF o che permettano di migrarle opportunamente. Inoltre il consumo di energia dell'infrastruttura di rete tradizionale può essere notevolmente ridotto grazie alla virtualizzazione, le operazioni di scalabilità aiutano a mettere in modalità di risparmio energetico i server quando la rete è scarica.

L'istanziazione e la configurazione di una VNF su un'infrastruttura deve avere delle prestazioni almeno equivalenti alla corrispondente PNF implementata su hardware dedicato. NFV si occupa di raccogliere le informazioni sulla computazione, memorizzazione e connettività per valutare le prestazioni della VNF. Un punto a favore dell'architettura NFV dal punto di vista prestazionale è la gestione di molte operazioni automaticamente grazie al NFV MANO. Le funzionalità di gestione possono essere fornite col modello informativo associato alla VNF e permettono di definire la capacità di adattamento della rete al carico, la configurazione e rilocazione, gli interventi nel caso venissero rilevati guasti, la gestione del ciclo di vita delle VNF (allocazione, scalabilità, terminazione). NFV colleziona ed analizza varie informazioni durante l'esecuzione dell'istanza VNF e li confronta con i vincoli e le politiche preimpostate.

I vari componenti che possono eseguire operazioni in parallelo potranno essere indicati nel modello informativo della VNF. NFV dà la possibilità di eseguire operazioni di scalabilità che possono essere innescate direttamente dalla VNF se si superano delle soglie prestabilite oppure possono essere avviate dal fornitore di servizi o dall'amministratore dell'architettura. Questa elasticità permette anche di migrare tutti i componenti della VNF da un ambiente ad un altro permettendo di offrire funzionalità di continuità del servizio.

NFV fornisce i meccanismi che permettono alle funzioni di rete di essere ricreate dopo un eventuale guasto. Il ripristino di un'istanza potrà avvenire automaticamente se nel modello informativo associato all'istanza erano stati specificati dei criteri di reazione al guasto oppure potrà avvenire manualmente se è l'amministratore dell'architettura a dover intervenire per ripristinare le istanze. L'affidabilità e la disponibilità di un'infrastruttura NFV verrà calcolata grazie alla raccolta di informazioni sul tasso di pacchetti persi, sul tasso di sessioni eliminate, sulla latenza, sulla variazione del ritardo e sul tempo necessario a rilevare e ripristinare la rete nel caso di un guasto.

L'introduzione della virtualizzazione porta a nuovi problemi di sicurezza che si devono prendere in considerazione:

- l'utilizzo dell'hypervisor introduce nuove vulnerabilità quindi si utilizzeranno hypervisor certificati da una terza parte fidata, verranno eseguiti su macchine aggiornate e contenenti solo il software essenziale per il suo funzionamento;

- la condivisione dell'hardware per la memorizzazione e la connettività aggiunge nuove vulnerabilità;
- la presenza di nuove interfacce fra i componenti dell'architettura NFV può portare a nuovi problemi di sicurezza;
- l'isolamento di una VNF dalle altre potrebbe non essere totale dunque questo è un altro campo in cui potrebbero essere introdotte delle vulnerabilità.

L'infrastruttura NFV fornisce dei meccanismi per implementare autenticazione, autorizzazione, cifratura e integrità dei dati, permette di creare dei ruoli associati alla possibilità di accedere ad un sottoinsieme delle funzionalità offerte ma alcuni dei problemi sopra elencati possono anche essere mitigati o risolti tramite l'utilizzo delle caratteristiche del Trusted Computing discusse nel Capitolo 2 con particolare riferimento al secure boot, al data sealing ed all'attestazione remota.

3.3.2 NFV MANO

Il compito principale del NFV MANO [29] è di gestire ed organizzare i vari componenti dell'architettura NFV ma scendendo più nel dettaglio esso svolgerà compiti differenti a seconda del componente a cui si rivolge, in particolare:

- gli aspetti coinvolti se si considera l'infrastruttura NFVI sono legati alle risorse fisiche o virtuali presenti. Verranno considerate sia le macchine fisiche sia quelle virtuali che offrono i servizi di computazione tramite la presenza di processori e memoria, di memorizzazione tramite volumi organizzati a blocchi o con un file system, di connettività tramite reti, sottoreti, porte, indirizzi e collegamenti. NFV MANO verifica e tiene traccia della disponibilità delle risorse, alloca le risorse nel caso in cui si debba istanziare una VNF, rilascia le risorse quando la VNF termina, rileva eventuali guasti ed adotta le contromisure necessarie al ripristino del servizio, colleziona i dati per analizzare le prestazioni;
- gli aspetti coinvolti se si considerano le VNF sono legati al ciclo di vita delle VNF. Essi includono la creazione di una VNF tramite l'utilizzo di un template che descrive i requisiti necessari a realizzare e gestire la VNF, le operazioni di scalabilità per aumentare o diminuire le capacità della VNF grazie all'utilizzo di apposite funzioni che permettono di monitorare le prestazioni della VNF, l'aggiornamento del software della VNF, la modifica dei parametri di configurazione della VNF, la terminazione della VNF, la gestione in caso di guasti, la raccolta delle informazioni relative alle risorse utilizzate, la gestione dei meccanismi di sicurezza e delle politiche associati alla VNF;
- gli aspetti coinvolti se si considerano i NS sono legati al ciclo di vita dei servizi di rete. Essi includono la necessità di assicurarsi che tutti i template che realizzano il NS siano stati caricati correttamente dal catalogo, la creazione di un NS partendo dal template caricato, l'esecuzione di operazioni di scalabilità grazie al monitoraggio delle prestazioni del NS, l'aggiornamento del NS ad esempio aggiungendo nuove VNF, la terminazione del NS, la gestione del VNFFG associato al NS con cui è possibile creare e modificare il flusso di traffico che attraversa le VNF.

La Figura 3.5 illustra in maggior dettaglio i componenti interni ad NFV MANO e quelli con cui esso interagisce, le funzionalità di alcuni di questi componenti sono già state discusse nella Sezione 3.3. Possiamo notare che NFV MANO gestirà dei cataloghi, in particolare:

- Catalogo NS: mantiene le informazioni per i servizi di rete caricati quindi conterrà una descrizione dei componenti che costituiscono il NS, un descrittore dei collegamenti realizzati e un descrittore di come fluisce il traffico all'interno del servizio;
- Catalogo VNF: mantiene le informazioni per le varie VNF caricate quindi conterrà una descrizione dei componenti che costituiscono la VNF e dell'immagine necessaria ad avviarla;

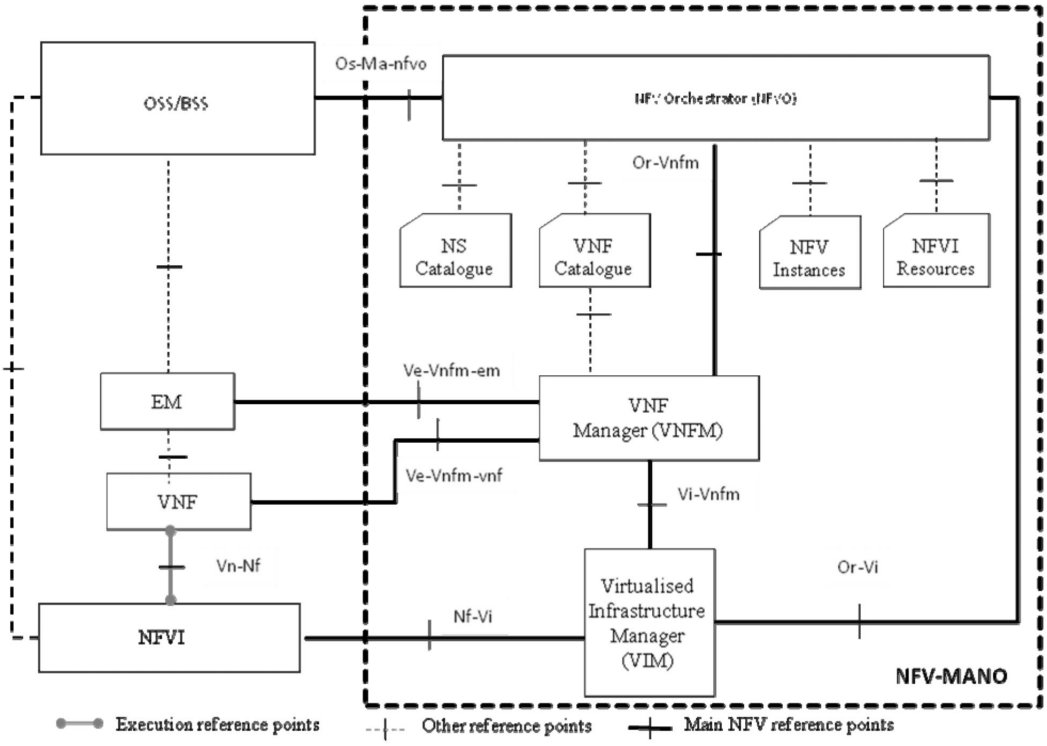

Figura 3.5. Visione dell'architetura NFV MANO (fonte:[29]).

- Repository delle istanze NFV: mantiene le informazioni sulle istanze VNF e sulle istanze NS. Durante il loro ciclo di vita verrà aggiornato un apposito record che riflette il loro stato attuale;
- Repository delle risorse NFVI: mantiene le informazioni sulle risorse NFVI disponibili, riservate ed allocate.

Gli elementi informativi permettono di descrivere dei dati presenti nei descrittori o nei record. La differenza fra queste due categorie è che i descrittori contengono delle informazioni statiche che servono a definire la VNF o il NS che dovrà essere creato mentre i record descrivono informazioni dinamiche che rispecchiano lo stato attuale dell'istanza VNF o NS. Ogni elemento informativo potrà contenere un solo valore ma potrebbe avere anche dei sotto elementi, la sua rappresentazione possiamo vederla come una tipica struttura ad albero in cui una foglia rappresenta un valore, un riferimento è un elemento che si riferisce a qualche altra informazione, un sottoelemento è un elemento informativo che specifica un altro livello nell'albero.

La rappresentazione di un NS può avvenire grazie alla definizione degli elementi informativi legati a Virtualised Network Function, Physical Network Function, Virtual Link e VNF Forwarding Graph. I descrittori che possiamo trovare in un'architettura NFV sono:

- Network Service Descriptor (NSD): descrive i componenti di un NS e verrà caricato nel catalogo NS;
- VNF Descriptor (VNFD): descrive lo sviluppo e il comportamento di una VNF e verrà caricato nel catalogo VNF;
- VNF Forwarding Graph Descriptor (VNFFGD): descrive il flusso di attraversamento delle VNF, PNF e link virtuali e verrà caricato nel catalogo NS;
- Virtual Link Descriptor (VLD): descrive i requisiti necessari a creare un collegamento fra VNF, PNF ed endpoint e verrà caricato nel catalogo NS;

- Physical Network Function Descriptor (PNFD) descrive la connettività della funzione fisica col collegamento virtuale.

Ci saranno dei record analoghi caricati per le istanze NFV quindi si parlerà di Network Service Record (NSR), VNF Record (VNFR), Virtual Link Record (VLR), VNFFG Record (VNFFGR) e Physical Network Function Record (PNFR).

Ogni descrittore o record può essere rappresentato tramite i suoi elementi e sottoelementi, esso deve rispettare delle caratteristiche ben precise ed è stato descritto dettagliatamente dall'ETSI nel documento [29].

Capitolo 4

Stato dell'arte ed obiettivo

In questo capitolo introdurremo lo scenario nel quale ci troviamo ad operare nella Sezione 4.1, approfondiremo i principali rischi di sicurezza e le contromisure da adottare nella Sezione 4.2, discuteremo gli obiettivi della tesi ed una possibile soluzione ad alto livello nella Sezione 4.3 ed introdurremo le principali caratteristiche dei software utilizzati nella Sezione 4.4.

4.1 Scenario

I numerosi vantaggi discussi nel Capitolo 3 stanno portando i fornitori di servizi a preferire un approccio di tipo NFV ma bisogna considerare alcuni aspetti che possono incidere nella sicurezza delle funzioni di rete sviluppate. Le prime applicazioni legate alla virtualizzazione prevedevano un server su cui potevano essere create delle macchine virtuali che permettessero di svolgere delle funzionalità l'una indipendentemente dall'altra ma sia i componenti hardware sia le macchine virtuali appartenevano ed erano gestite dalla stessa azienda. L'introduzione del paradigma NFV cambia questo approccio. Le risorse fisiche possono essere distribuite su più server e tutta l'architettura può essere gestita da aziende differenti, ad esempio un'azienda potrebbe gestire la parte infrastrutturale e l'altra si potrebbe occupare dell'orchestrazione e dello sviluppo dei servizi di rete virtuali. Questo approccio introduce delle nuove vulnerabilità. Il fornitore di servizi non sa esattamente dove verrà eseguito il servizio cioè in quale server fisico verrà creata la macchina virtuale. In generale chi gestisce i servizi di rete dovrebbe avere piena fiducia sulle politiche di sicurezza adottate dall'organizzazione che gestisce l'infrastruttura. Bisogna considerare anche il caso in cui un utente malevolo riesca a compromettere il server fisico su cui gira il servizio di rete e nelle condizioni normali di utilizzo chi usufruisce del servizio di rete non se ne accorgerebbe. Serve un meccanismo che permetta di verificare lo stato del server e che permetta di avviare il servizio solo se il server viene ritenuto fidato. È necessario, dunque, un meccanismo che permetta di controllare che la macchina virtuale su cui è in esecuzione il servizio di rete è stata creata su di un server considerato fidato, per questo scopo può essere sfruttato il Trusted Computing discusso nel Capitolo 2 ed in particolare il meccanismo di attestazione attraverso cui è possibile riportare in maniera sicura dei dati legati alla piattaforma per verificare la genuinità del software che gira su quel server.

La Figura 4.1 mostra il differente approccio fra un'architettura che non utilizza l'attestazione remota ed una che la utilizza, in particolare si nota che nel primo caso il servizio di rete viene creato su un server che è già stato compromesso e l'utilizzatore del servizio non se ne accorge mentre nel secondo caso il meccanismo di attestazione permette di evitare il server compromesso e di avviare il servizio di rete su un server fidato.

4.2 Rischi di sicurezza

L'analisi svolta da Shankar Lal, Tarik Taleb e Ashutosh Dutta nell'articolo [30] permette di identificare numerose minacce che si possono manifestare in un'architettura NFV e propone delle linee

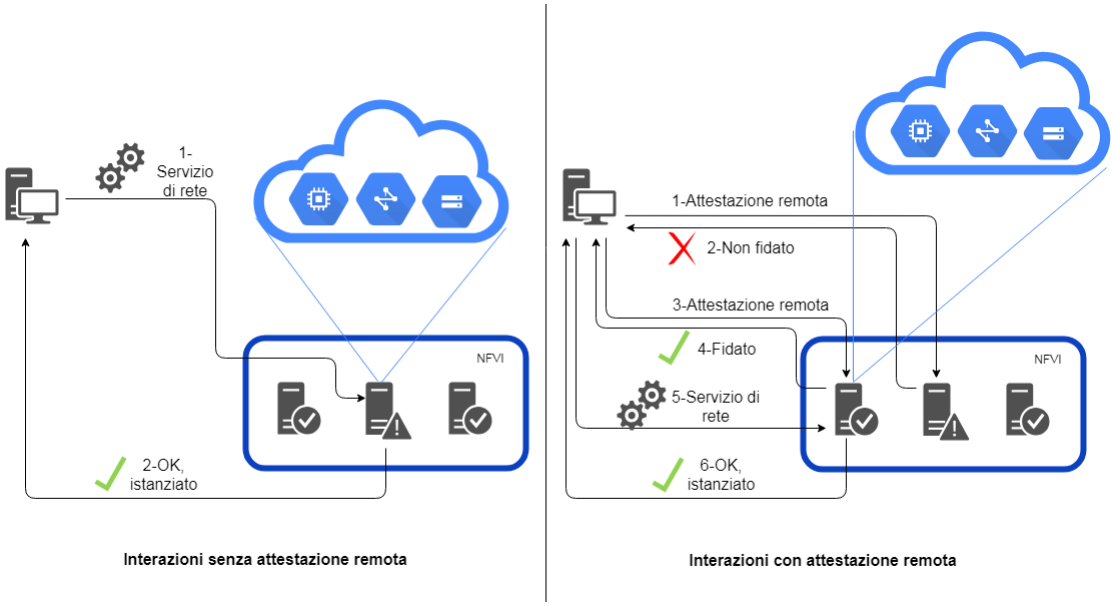

Figura 4.1. Scenario senza e con attestazione remota.

guida per una corretta configurazione dei dispositivi. Le minacce di sicurezza associate alle VNF sono la combinazione di quelle che potrebbero colpire la rete fisica e di quelle legate alla tecnologia di virtualizzazione. Possibili rischi di sicurezza sono:

- isolamento insufficiente: è legato all'isolamento fra hypervisor e VNF. Se l'isolamento non fosse ben strutturato un utente malevolo potrebbe compromettere la VNF e tramite essa raggiungerebbe l'hypervisor, prendendo possesso di esso avrebbe la possibilità di creare, modificare o eliminare le VNF;
- difetti nell'implementazione: la cattiva configurazione di router virtuali e firewall può portare ad un'impropria separazione fra reti e sottoreti ed un attaccante potrebbe sfruttarlo per avere la piena conoscenza dell'infrastruttura;
- mancato rispetto delle norme: l'architettura NFV permette di spostare le VNF in luoghi differenti. Un utente malevolo potrebbe forzare la migrazione delle VNF in luoghi ritenuti illegali. La violazione delle leggi potrebbe comportare per il fornitore di servizi delle sanzioni pecuniarie e il blocco totale all'esercizio delle funzionalità di fornitore di servizi;
- Denial of Service (DoS): è un attacco che potrebbe essere diretto alle VNF o alla rete virtuale ed ha conseguenze sulla disponibilità del servizio. Ad esempio, un attaccante ruba degli IP validi per quella rete a degli utenti vittima e genera numerose query al DNS, l'orchestratore per gestire il carico avvia nuovi DNS virtuali e tutte le risposte inondano le vittime saturando le loro risorse.

Le migliori pratiche di sicurezza in un'architettura NFV prevedono:

- la misurazione dei componenti sensibili nei sistemi che costituiscono l'infrastruttura fisica NFV. Questi sistemi, nella fase di avvio, utilizzeranno il TPM come root of trust hardware per misurare ed immagazzinare in maniera sicura i dati relativi al firmware, al BIOS, al kernel e ad altri componenti. In seguito sarà possibile validare queste misure con la procedura di attestazione remota;
- l'hypervisor deve essere regolarmente aggiornato e bisogna disabilitare tutti i servizi che non si utilizzano. Ad esempio, SSH si terrà normalmente disabilitato e si abiliterà solo quando serve;

- l'isolamento del traffico che transita sull'infrastruttura è un requisito necessario per separare i dati delle VM dal traffico associato alla loro gestione. Bisogna anche raggruppare le VM in apposite regioni separate dalle altre per evitare che eventuali problemi si ripercuotano su tutta l'infrastruttura;
- l'introspezione dell'hypervisor che permette di controllare il software dell'hypervisor per rilevare eventuali attività anomale;
- i volumi associati alle VNF possono contenere dati sensibili quindi devono essere protetti attraverso la loro cifratura e la memorizzazione delle chiavi crittografiche in luoghi sicuri;
- la firma sulle immagini VNF. Per un utente malevolo sarebbe semplice creare un'immagine VNF con un malware e caricarla nel database delle immagini quindi è necessario firmare digitalmente ogni immagine in fase di caricamento nel database e verificare la firma nella fase di lancio;
- l'attestazione remota che permette di verificare lo stato di fiducia di una piattaforma NFV.

Queste linee guida devono essere tenute in considerazione nello sviluppo di una soluzione che permetta di gestire i servizi di rete in un'architettura NFV con un certo livello di sicurezza, alcuni di questi aspetti hanno avuto un maggior rilievo durante il lavoro di tesi.

L'autenticazione e l'attestazione sono due elementi complementari, necessari nella realizzazione di un ambiente NFV sicuro [31]. L'autenticazione permette di verificare l'identità di una piattaforma, l'attestazione permette di verificare se la piattaforma è fidata calcolando delle informazione sul suo stato di integrità. L'attestazione avviene all'avvio della macchina e potrà essere ripetuta periodicamente per mantenere il sistema sotto controllo. I tentativi di manomissione del sistema sono rilevati in breve tempo grazie a questo meccanismo. Lo stato di fiducia di una piattaforma si calcola utilizzando le cosiddette root of trust, discusse già nel Capitolo 2, cioè delle primitive implementate in hardware e ritenute fidate già in partenza. Esse forniscono la protezione del materiale crittografico, l'isolamento delle operazioni crittografiche e l'avvio del codice per le misurazioni. Una catena di fiducia viene realizzata grazie alle funzioni che permettono di misurare i componenti del sistema, di comunicare in maniera sicura queste misurazioni e di memorizzarle. Il processo di misurazione all'avvio assicura che tutti i componenti siano misurati prima della loro esecuzione e che tali misure siano registrate in un file di log protetto tramite l'integrità.

Le funzionalità di secure boot e trusted boot possono essere usate entrambe ed in maniera complementare. Il secure boot permette di verificare la firma digitale dei principali componenti presenti sulla macchina all'avvio, se per qualcuno di questi componenti la firma non viene verificata allora si impedisce il normale avvio del sistema e si fa partire una procedura di recupero. Il trusted boot permette di raccogliere e memorizzare le misure eseguite sui vari componenti durante la fase di avvio, tali misure successivamente verranno inviate ad un verificatore che potrà controllare lo stato attuale della piattaforma.

La necessità dei fornitori di servizi e dei gestori di risorse è di sapere se il loro servizio e quindi le VNF che lo implementano sono sicure.

4.3 Obiettivo

I fornitori di servizi di rete stanno migrando le loro architetture verso la piena compatibilità con NFV quindi i componenti che costituiscono l'architettura possono appartenere a diversi livelli di compatibilità [32]:

- livello 0: specifica le funzioni di rete tradizionali implementate con hardware e software proprietario;
- livello 1: specifica le funzioni di rete implementate direttamente su server COTS quindi non virtualizzate;

- livello 2: specifica le funzioni di rete virtualizzate singolarmente su un server COTS;
- livello 3: specifica le funzioni di rete virtualizzate ed orchestrate insieme ad altre funzioni di rete per ottimizzare le risorse fisiche cioè i server COTS non sono dedicati ad una specifica funzione;
- livello 4: specifica le funzioni di rete virtualizzate e gestite completamente dall'orchestratore che si occupa del ciclo di vita, della scalabilità e dei guasti delle VNF.

Nell'ottica che i vari fornitori di servizi hanno intenzione di migrare le loro strutture verso il livello 4, è necessario permettere all'orchestratore anche la gestione delle caratteristiche di sicurezza. L'obiettivo della tesi è di estendere le funzionalità dell'orchestratore per supportare l'attestazione remota cioè l'orchestratore ha la possibilità di sapere se il server su cui sta per avviare un servizio di rete può essere considerato fidato, se le macchine virtuali che costituiscono il servizio sono fidate e in caso di problemi può decidere di bloccare completamente l'avvio del servizio o di avviarlo ugualmente segnalando la pericolosità dell'evento, ciò dipende dalle politiche che hanno stabilito il gestore dei servizi ed il gestore della sicurezza.

Una visione ad alto livello della struttura architetturale che potrebbe risolvere questo problema prevede l'utilizzo e la cooperazione dell'architettura che implementa l'approccio ETSI NFV e di quella che permette di gestire le caratteristiche legate all'attestazione; l'orchestratore avrà la possibilità di verificare lo stato dei server e di decidere se avviare il servizio su di esso.

La Figura 4.2 mostra le interazioni dell'architettura di attestazione con l'infrastruttura NFV

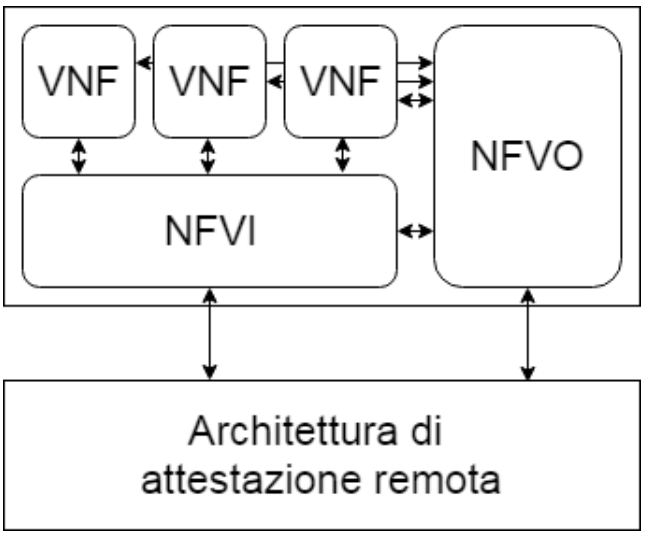

Figura 4.2. Soluzione ad alto livello.

ai fini della verifica dello stato di integrità dei server e quelle con l'orchestratore per riportare le informazioni legate all'attestazione.

4.4 Software utilizzati

L'obiettivo di realizzare un'architettura in cui sia possibile avviare un servizio di rete in maniera sicura ci ha portato ad analizzare alcuni software che permettono di implementare le funzionalità adatte al nostro scenario.

OpenStack

OpenStack [33] è un progetto open source nato nel 2010 dalla collaborazione fra Rackspace Cloud e National Aeronautics and Space Administration (NASA). OpenStack può essere definito un

Infrastructure as a Service (IaaS) cioè esso fornisce l'accesso ad un'infrastruttura virtualizzata ed ha la possibilità di allocare o deallocare risorse fisiche in base alla reale necessità della piattaforma. OpenStack fornisce computazione, memorizzazione e connettività. Esso può essere visto come il sistema operativo del cloud perché permette di astrarsi dalle risorse fisiche realmente utilizzate fornendo i servizi necessari al suo corretto funzionamento così come un classico sistema operativo si occupa di gestire le risorse dell'elaboratore su cui è installato in maniera trasparente rispetto all'utente che lo sta utilizzando (gestisce lo scheduling dei processi da mandare in esecuzione, gestisce gli indirizzi di memoria, gestisce le comunicazioni fra le periferiche presenti).

I principali vantaggi [34] che offre OpenStack sono:

- elasticità: la possibilità di allocare e deallocare risorse velocemente e dinamicamente senza l'intervento di un operatore;
- interoperabilità: la possibilità di utilizzare elementi fisici di diversi produttori grazie all'uso di driver che permettono la corretta interazione;
- scalabilità: la possibilità di aumentare o diminuire le risorse in base al carico presente;
- tolleranza ai guasti: la possibilità di far proseguire il servizio in esecuzione isolando la zona su cui si verifica il guasto;
- multiutenza: la possibilità di fornire servizi a più utenti contemporaneamente.

La Figura 4.3 illustra i principali moduli di cui si compone l'architettura OpenStack. Possiamo

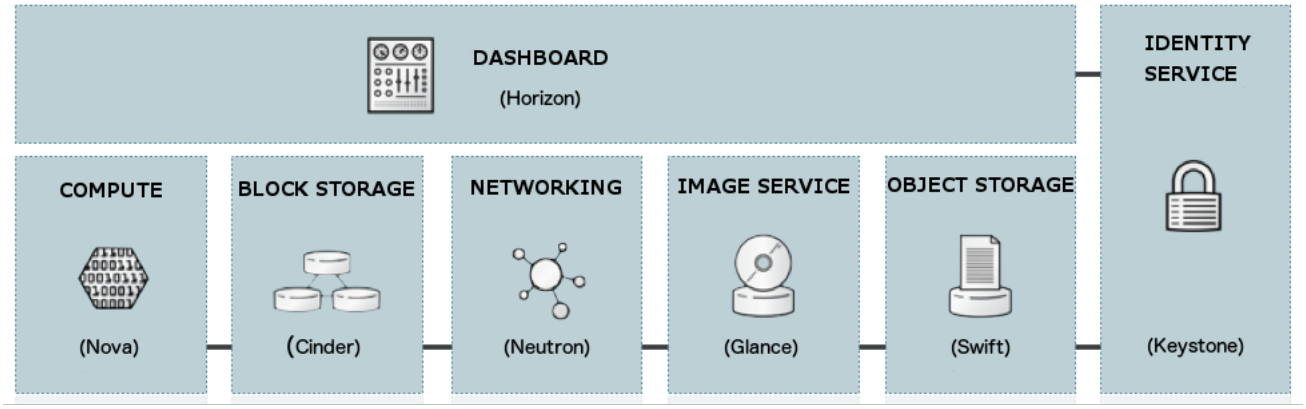

Figura 4.3. Componenti OpenStack (fonte:[33]).

notare come la Dashboard e l'Identity Service siano i componenti fondamentali che si interconnettono con tutti gli altri. Ogni servizio potrà essere eseguito solo se il servizio che gestisce le identità lo ha autenticato. Ogni servizio mostrerà le sue funzionalità tramite l'interfaccia grafica presente nella Dashboard. Ogni servizio esporrà delle proprie API pubbliche attraverso cui tutti gli altri possono interagire. Le principali funzionalità ricoperte dai moduli OpenStack sono:

- Identity service (Keystone): fornisce i servizi di autenticazione ed autorizzazione all'intera infrastruttura cloud. Riesce a gestire varie forme di autenticazione come quella basata su nome utente e password o quella basata su sistemi a più fattori. Fornisce un catalogo dei servizi disponibili nell'infrastruttura.
- Image service (Glance): fornisce il servizio di memorizzazione e gestione per le immagini con cui si avvieranno le macchine virtuali.
- Block Storage service (Cinder): fornisce un servizio di memorizzazione persistente. Permette di creare dischi virtuali da aggiungere alle macchine virtuali così da estendere la quantità di spazio di archiviazione a loro disposizione.

- Object Storage service (Swift): fornisce il supporto per memorizzare i dati o le immagini delle macchine virtuali e si occupa della loro replica per fornire maggiore affidabilità in caso di guasti. Offre un servizio distribuito, scalabile, ad alta disponibilità ed affidabilità.
- Compute service (Nova): implementa le funzionalità di computazione e permette di gestire le istanze virtuali grazie all'hypervisor attraverso cui è possibile astrarsi dalle risorse fisiche sottostanti. OpenStack non include il software di virtualizzazione ma fornisce dei driver per comunicare con i meccanismi di virtualizzazione dell'host, in sistemi Linux uno dei più comuni hypervisor utilizzati è KVM.
- Networking service (Neutron): viene definito un Network as a Service (NaaS) poiché fornisce delle API per supportare la connettività di rete e l'indirizzamento nel cloud. Fornisce l'astrazione di alcuni elementi che è possibile trovare in un'infrastruttura reale come i router o il DHCP.
- Dashboard (Horizon): fornisce un'interfaccia grafica sia per gli amministratori sia per gli utenti. Permette di gestire e monitorare le risorse del cloud fornendo un accesso a tutti i servizi sviluppati in OpenStack.
- Key Manager service (Barbican): gestisce e permette di memorizzare in maniera sicura elementi come chiavi simmetriche ed asimmetriche, certificati o dati binari.

L'architettura OpenStack potrebbe essere realizzata su un'unica macchina fisica ma non è la soluzione migliore in ambito produttivo. Per poter sfruttare a pieno le potenzialità di OpenStack sono necessarie almeno due macchine fisiche distinte. Una macchina svolgerà il compito di controller node cioè conterrà il software necessario per pilotare e gestire tutte le funzionalità offerte da OpenStack. L'altra macchina svolgerà il compito di compute node cioè conterrà il software necessario ad eseguire i calcoli e permetterà l'avvio su di essa delle macchine virtuali.

Open Source MANO

Open Source MANO (OSM) [35] è un progetto ospitato dallo European Telecommunications Standards Institute (ETSI). Si occupa di sviluppare il software necessario ad implementare il componente per la gestione e l'orchestrazione delle risorse presente nell'architettura NFV noto col nome di MANO. OSM è totalmente compatibile ed allineato con le specifiche ETSI NFV.

I punti chiave che caratterizzano questo software sono:

- la modellazione semplificata sia delle VNF sia dei servizi di rete grazie all'utilizzo di un'interfaccia grafica che permette di convertire il grafico nei corrispondenti descrittori;
- il supporto a Cloud-init [36] che permette di iniettare la configurazione iniziale nelle macchine virtuali;
- i descrittori vengono rappresentati tramite YAML (YAML Ain't a Markup Language) [37] che è un linguaggio adatto alla serializzazione dei dati;
- i VIM supportati sono OpenVIM [38], OpenStack, VMware vCloud Director [39] e Amazon web service EC2 [40];
- la possibilità di aggiungere o rimuovere delle VNF da un servizio di rete in esecuzione;
- la presenza di un client gestibile tramite linea di comando permette di interagire con OSM tramite API REST;
- il ripristino veloce del servizio di rete in caso di guasto dei singoli componenti.

L'architettura OSM è costituita principalmente da tre componenti:

1. Resource Orchestrator (RO): è l'elemento che interagisce col VIM sottostante. Il VIM sarà un elemento esterno all'architettura OSM. RO si occupa di tradurre i comandi che riceve dal livello superiore in comandi adatti al VIM. Permette di gestire ed allocare le risorse per implementare un servizio di rete.

2. VNF Configuration and Abstraction (VCA): offre alcune funzionalità del VNFM del modello ETSI NFV. Esso permette di inserire la configurazione iniziale nelle VNF e di gestire il ciclo di vita dei servizi di rete. Attualmente OSM supporta la configurazione *day 0* cioè quella inserita prima di avviare la macchina oppure la configurazione *day 1* cioè quella che viene iniettata una volta che la macchina è stata avviata ad esempio sfruttando un meccanismo con chiavi SSH o delle API che erano già state predisposte per ricevere la configurazione.

3. Network Service Orchestrator (NSO): espone un'interfaccia grafica per comunicare con l'utente ed è il componente responsabile dello sviluppo del NS. Esso gestisce il catalogo di VNF e NS all'interno del quale è possibile inserire i descrittori, le immagini da avviare e gli script di configurazione. Permette la validazione dei descrittori utilizzando il modello YANG (Yet Another Next Generation) [41].

OSM permette di esprimere i descrittori in formato YAML e di validarli attraverso YANG. YANG è un linguaggio che permette di definire un modello dati per i dispositivi di rete. Esso riesce a validare qualsiasi linguaggio che permetta di serializzare dei dati come XML o JSON o YAML. I descrittori permettono di rappresentare degli elementi informativi in OSM. Essi mantengono informazioni associate ai servizi di rete ed alle VNF. OSM supporta varie tipologie di descrittori:

- il descrittore del servizio di rete (NSD) rappresenta le varie caratteristiche associate ad un servizio di rete come le VNF di cui è costituito, i collegamenti fra le varie VNF e come fluiscono i dati al suo interno. NSD sarà costituito da vari descrittori che permettono di rappresentarlo;

- il descrittore della VNF (VNFD) contiene tutte le informazioni sulla VNF come il tipo di CPU utilizzata, la quantità di memoria, le interfacce e le reti presenti, le proprietà di scalabilità, i collegamenti virtuali;

- il descrittore dei collegamenti virtuali (VLD) definisce i collegamenti fra le VNF sia a livello data link sia a livello di rete. Permette di specificare il tipo di collegamento che può essere esterno se permette la connessione al servizio di rete o interno se connette le VNF fra loro;

- il descrittore del grafo di inoltro nelle VNF (VNFFGD) contiene dei metadati associati alle VNF ed ai loro collegamenti. Si basa su politiche e regole che permettono di determinare il flusso di traffico del servizio di rete.

Open CIT

Open Cloud Integrity Technology (Open CIT) [42] è una tecnologia sviluppata da Intel che permette di eseguire l'attestazione remota in ambiente cloud. Come visto nel Capitolo 2, quando avviamo un sistema è necessaria la presenza di componenti ritenuti fidati detti root of trust per poter creare una catena di fiducia e grazie a questo meccanismo sarà possibile eseguire, in seguito, l'attestazione remota della piattaforma.

Open CIT permette di estendere le root of trust anche in ambiente cloud grazie alla presenza di un'architettura che assicuri le proprietà di privacy ed integrità alle varie piattaforme. Un requisito necessario nelle piattaforme su cui dovrà essere eseguita l'attestazione remota è la presenza e l'abilitazione della tecnologia Intel TXT (Trusted Execution Technology) [43]. Essa sfrutta la presenza del TPM per avviare il sistema in maniera fidata abilitando l'isolamento e il rilevamento di eventuali manomissioni ed eseguendo la verifica direttamente a livello hardware. Le applicazioni controlleranno lo stato della piattaforma e le politiche impostate prima di avviare qualsiasi carico nel sistema. Intel TXT, all'avvio, si occupa di misurare sia il firmware del server sia il software presente così ogni componente aggiuntivo come hypervisor, macchine virtuali o contenitori Docker

potrà essere attestato e verificato. È possibile creare una catena di fiducia infatti ad ogni passo della fase di avvio verranno misurati i componenti del passo successivo. L'architettura Open CIT prevede la presenza di un Attestation Server per verificare i valori misurati e per confrontarli con i valori di riferimento precedentemente memorizzati. I valori che vengono presi in considerazione nella fase di misurazione sono il firmware del BIOS, il kernel, l'hypervisor, i file e le cartelle scelti dall'amministratore. Le principali funzionalità [44] che caratterizzano Open CIT sono:

- Trusted Boot e Trusted Pools: permettono di avere la prova che il server su cui si andrà ad avviare una macchina virtuale è fidato;
- Trusted Geo Tag e Trusted Asset Tag: permettono di localizzare i server su cui eseguire il carico di lavoro;
- Tenant Controlled Protected e Trusted Workloads: permette di proteggere il carico di lavoro utilizzando la cifratura e verificando lo stato fidato dei server;
- Run Time Protection: assicura che il carico di lavoro e i dati siano protetti durante l'esecuzione.

La Figura 4.4 mostra i principali componenti di Open CIT:

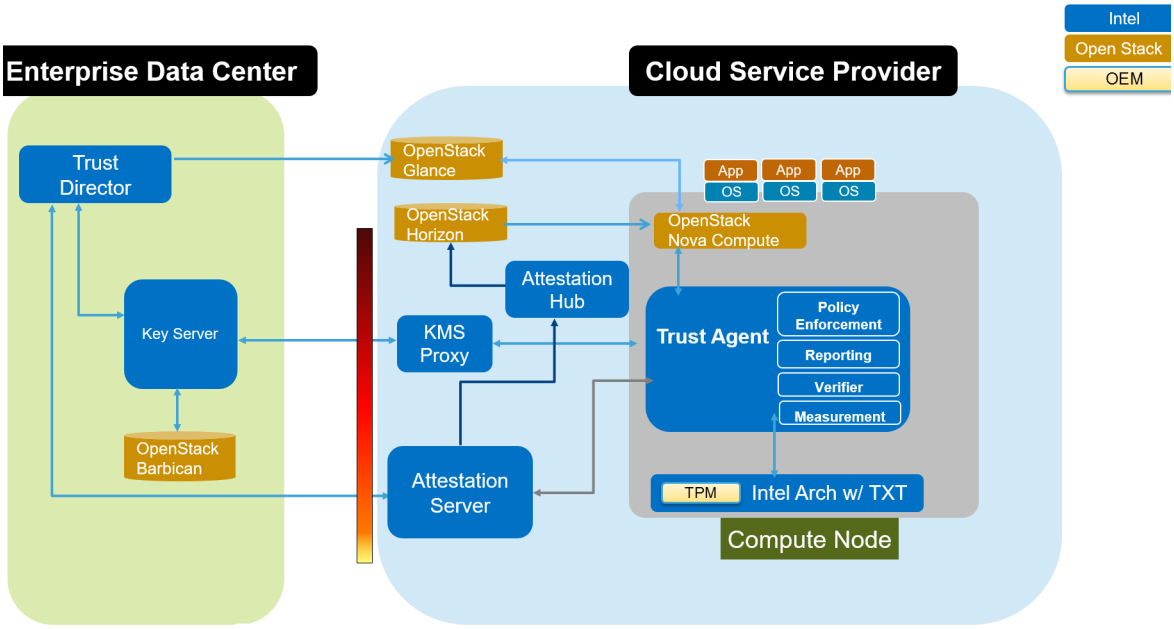

Figura 4.4. Architettura Open CIT (fonte:[45]).

- Attestation Server: esegue l'attestazione remota dei server fisici confrontando le misurazioni relative al BIOS, al sistema operativo e agli altri componenti con i valori di riferimento che erano già stati memorizzati nel database.
- Key Broker Server (KMS): abilita la distribuzione delle chiavi usate per cifrare e decifrare le immagini delle macchine virtuali. La creazione e gestione delle chiavi non è gestita direttamente da KMS ma si appoggia ad un servizio esterno che nel caso di OpenStack è Barbican.
- Key Broker proxy (KMS proxy): è un servizio che serve a migliorare le prestazioni. Fa da intermediario fra il KMS e il Trust Agent;

- Trust Director (TD): ha il compito di generare la trust policy cioè un file che contiene le informazioni relative agli elementi su cui andare ad eseguire l'attestazione. TD ha la possibilità di creare trust policy non virtualizzate o virtualizzate. La trust policy non virtualizzata viene aggiunta nel file system del server fisico e nella fase di avvio permetterà di estendere la catena di fiducia. La trust policy virtualizzata viene caricata insieme all'immagine della macchina virtuale sul servizio OpenStack Glance e potrà essere di tipo:
  - hash only: il Trust Agent controlla lo stato dell'integrità dell'immagine rispetto la propria trust policy ed avvia la macchina virtuale indipendentemente dal risultato;
  - hash and enforce: il Trust Agent controlla lo stato dell'integrità dell'immagine rispetto la propria trust policy ed impedisce l'avvio della macchina virtuale se il risultato è non fidato.
- Trust Agent: viene installato nel server fisico che dovrà essere attestato e abilita le capacità di attestazione remota e di estensione della catena di fiducia. Esso riesce ad interagire col TPM nel server.
- Attestation Reporting Hub: agisce come intermediario fra l'Attestation Server e i servizi di scheduling OpenStack. Esso richiede periodicamente le informazioni di attestazione all'Attestation Server e le inserisce nello scheduler OpenStack.

Capitolo 5

Soluzione realizzata

In questo Capitolo presenteremo la soluzione realizzata per risolvere i problemi esposti nel Capitolo 4. La Sezione 5.1 analizzerà le motivazioni che ci hanno portato alla scelta dei software utilizzati durante la tesi. La Sezione 5.2 mostrerà nel dettaglio l'architettura realizzata ed i relativi componenti. La Sezione 5.3 illustrerà come i componenti appartenenti ai vari software possano coesistere ed interagire fra loro. La Sezione 5.4 descriverà il componente di Open Source MANO che è stato esteso. La Sezione 5.5 illustrerà il diagramma di flusso rappresentativo dell'algoritmo implementato come soluzione. La Sezione 5.6 mostrerà il flusso di esecuzione della soluzione realizzata.

5.1 Scelta dei software

Il primo problema da affrontare è la scelta dei software necessari alla realizzazione di una soluzione che permetta di realizzare un'infrastruttura virtualizzata, di orchestrare adeguatamente le sue risorse e di rafforzare la sicurezza con meccanismi che permettano di verificare lo stato di integrità dei server fisici su cui verrà sviluppato un servizio di rete. Il laboratorio in cui è stata implementata la soluzione offre delle macchine con CPU Intel quindi la scelta più logica è stata quella di sfruttare le tecnologie offerte dalla Intel come Intel Trusted Execution Technology (TXT). Questi, utilizzato in cooperazione col TPM, permette la costruzione della catena di fiducia. Open CIT è il software necessario per assicurarsi che i servizi di rete vengano eseguiti su server fisici affidabili ed inalterati. Come mostrato nel manuale in Appendice A, uno dei vincoli per l'installazione e l'utilizzo di Open CIT è la presenza di OpenStack come IaaS. Il fornitore di servizi cloud dovrà utilizzare tale software grazie a cui sono offerte: affidabilità, scalabilità, presenza di database distribuiti, modularità, supporto a vari tipi di hypervisor per lo sviluppo di macchine virtuali, aggiornamenti periodici, ampia comunità di supporto, possibilità di modificare il codice sorgente del software, nessuna licenza necessaria per poterlo utilizzare. OpenStack insieme all'hypervisor KVM sono considerati gli standard de facto dalla comunità open source. Come mostrato nella presentazione di G. Lavado all'OpenStack summit di Vancouver del 2018 [46] più del 70% delle installazioni utilizza OpenStack come VIM.

La scelta del software di orchestrazione ha un ventaglio di opzioni più ampio [47], sicuramente è meglio analizzare software open source perché tale tipologia permette di essere indipendenti da uno specifico produttore e velocizza i tempi per integrare soluzioni innovative. Esistono numerosi software realmente funzionanti che possono contare su una comunità di supporto attiva.

Open Source MANO (OSM) è stato discusso nel Capitolo 4. È un progetto lanciato nel 2016 da ETSI, è allineato con le specifiche ETSI NFV MANO ed inoltre include la collaborazione di numerose aziende. L'architettura è costituita principalmente da tre componenti:

- Network Service Orchestrator: interagisce con l'utente e gestisce l'orchestrazione del servizio;
- Resource Orchestrator: interagisce col VIM per lo sviluppo dei servizi di rete;

- VNF Configuration and Abstraction: gestisce la configurazione iniziale delle VNF.

Questi componenti sono dei contenitori creati utilizzando la tecnica di virtualizzazione leggera LXD (linux deamon) [48] che si basa su LXC (linux container) [49]. La virtualizzazione leggera ha il vantaggio di creare i contenitori direttamente nel sistema operativo ospitante (il sistema operativo fungerà anche da hypervisor) garantendo maggiore velocità di avvio e di esecuzione rispetto ad una classica macchina virtuale e fornendo il controllo delle risorse e l'isolamento tramite caratteristiche già presenti nel kernel Linux. OSM segue le specifiche ufficiali per definire i descrittori dei servizi di rete e delle VNF, supporta più VIM (attualmente OpenVIM, OpenStack, VMware e AWS) ma il VNFM non supporta ancora tutte le funzionalità richieste da ETSI NFV. L'ampia comunità che lo supporta mantiene alto il livello di interesse su questo software creando eventi in cui rendere pubbliche le nuove funzionalità implementate e mettendo in contatto diretto gli sviluppatori con gli utilizzatori del software, numerose aziende supportano questo prodotto con contributi economici.

OpenBaton [50] è un software sviluppato dall'Università di Berlino che ha come obiettivo lo sviluppo di un'architettura capace di orchestrare i servizi di rete in infrastrutture NFV eterogenee. Questo software permette di gestire le VNF attraverso un VNFM generico che è allineato con le specifiche ETSI NFV. Sarebbe possibile anche integrare un proprio VNFM sfruttando le REST API messe a disposizione dal NFVO per interagire con esso. OpenBaton supporta numerosi VIM tuttavia OpenStack è quello maggiormente utilizzato. Gli elementi principali di OpenBaton sono:

- Network Function Virtualisation Orchestrator (NFVO): rispecchia le funzionalità di orchestrazione descritte nelle specifiche ETSI;
- Virtual Network Function Manager (VNFM) e Element Management System (EMS) generici: sono gli elementi capaci di gestire il ciclo di vita delle VNF;
- adattatore Juju VNFM: permette di sviluppare Juju charm;
- meccanismo che supporta differenti tipi di VIM, attualmente sono supportati OpenStack e AWS;
- driver per istanziare contenitori Docker;
- motore di autoscaling esterno che permetta di gestire in maniera automatica le operazioni di scalabilità delle VNF durante la loro esecuzione;
- sistema per gestire i guasti;
- librerie per costruire il proprio VNFM.

OpenBaton è un software tecnicamente ben organizzato e strutturato ma ha una comunità di supporto limitata come mostrato nel documento [46].

Open Network Automation Platform (ONAP) [51] fu lanciato nel 2017 dalla Linux Foundation come l'unione di altri due progetti che erano OpenECOMP ed OPEN-O. L'obiettivo di ONAP è realizzare una piattaforma per l'orchestrazione e l'automazione di funzioni di rete fisiche e virtuali attraverso cui gestire completamente il loro ciclo di vita. Si possono individuare degli elementi in comune con l'architettura ETSI NFV MANO anche se questo software non si pone come obiettivo il raggiungimento della piena compatibilità con le specifiche ETSI. L'architettura è costituita principalmente da due macro componenti che sono l'ambiente di progettazione (raccoglie un insieme di tool necessari a creare dei servizi che sarà possibile avviare nell'architettura) e l'ambiente di esecuzione (contiene un insieme di componenti per l'avvio, il controllo e la gestione dei servizi). Più nel dettaglio gli elementi di cui è costituito sono:

- Service Design and Creation (SDC): fa parte dell'ambiente di progettazione e permette agli sviluppatori di definire, simulare e certificare i componenti da avviare e le politiche associate ad essi. Le politiche permettono di creare delle regole relative ai requisiti, ai vincoli, agli attributi di ciò che deve essere gestito in quest'architettura.

- Active and Available Inventory (AAI): crea una visione globale della topologia di rete e la mantiene aggiornata durante l'esecuzione.
- Controller: è presente il controller della rete che si occupa di istanziare le VNF e controllare il loro stato e il controller delle applicazioni che ottiene i componenti e gli attributi dal SDC e si occupa di configurare le loro risorse.
- Dashboard: fornisce un'interfaccia grafica per accedere alle varie funzionalità di progettazione, gestione ed istanziazione delle applicazioni.
- Data Collection, Analytics and Event (DCAE): colleziona i dati associati alle configurazioni ed alle esecuzioni per fornire un'analisi sulle prestazioni.
- Master Service Orchestrator (MSO): gestisce l'orchestrazione ad alto livello, traduce ed invia i dati agli opportuni controller sottostanti per avviare un servizio.
- Virtual Function Controller (VFC): gestisce il ciclo di vita di VNF e servizi di rete.
- Common service: gestisce servizi utilizzati dai vari componenti come l'ottimizzazione delle politiche per la creazione delle applicazioni, il controllo degli accessi o il tracciamento di ciò che accade nel sistema.

ONAP offre il supporto a molti VIM fra cui OpenStack e VMware, è un software tecnicamente completo e molto ampio, ha una grande comunità di supporto ma non sembra ancora pronto ad un utilizzo vero e proprio.

Cloudify [52] è stato inizialmente creato da GigaSpaces come pura soluzione di orchestrazione ma è stato esteso per includere elementi ETSI NFV, supporta molti IaaS (AWS, OpenStack, Azure, etc), ed è un progetto open source. Lo scopo principale del progetto è l'orchestrazione di applicazioni nel cloud. Esso implementa NFVO e VNFM secondo le specifiche ETSI NFV ma non implementa alcune interfacce ed azioni quindi si può considerare allineato ad ETSI NFV ma non pienamente compatibile. L'architettura è costituita dal *cloudify manager* che gestisce lo sviluppo ed il ciclo di vita dei servizi di rete, dai *cloudify agent* che sono presenti su ogni host e gestiscono la coda di processi eseguendoli quando è necessario, dalla *cloudify console* che è l'interfaccia da linea di comando attraverso cui è possibile impartire dei comandi. I principali componenti presenti nell'architettura sono:

- Logstash: processa le informazioni di log che vengono dai componenti della piattaforma e dagli agent;
- Riemann: durante l'esecuzione prende decisioni relative alla disponibilità dei componenti e permette di monitorare il loro comportamento;
- RabbitMQ: gestisce la coda di messaggi attraverso cui i componenti della piattaforma possono comunicare fra loro;
- Nginx: è un server web che funge da proxy per i servizi REST cloudify e mantiene le risorse dei componenti nell'host;
- PostgreSQL: è un database che mantiene le informazioni relative al carico di lavoro delle singole macchine e permette il loro utilizzo nelle applicazioni.

Cloudify permette di istanziare un servizio utilizzando dei modelli detti blueprints che contengono i piani di orchestrazione e sviluppo cioè verrà descritta la topologia del servizio e potranno essere aggiunte delle risorse esterne come script per la configurazione e l'installazione. Questo software supporta numerosi VIM (OpenStack, VMware, Azure, etc) ed altri possono essere aggiunti semplicemente aggiungendo il loro plugin, inoltre è presente il supporto a servizi multisito ma sono necessari plugin specifici per poterli realizzare.

Allo stato attuale Open Source MANO sembra essere il software più maturo e con il maggior supporto oltre ad essere totalmente compatibile con le specifiche ETSI NFV quindi sarà quello che verrà utilizzato come orchestratore nel lavoro di tesi.

5.2 Architettura

La soluzione proposta prevede un'architettura che permetta la connessione e cooperazione dei componenti che costituiscono i singoli software utilizzati. La Figura 5.1 mostra dal punto di

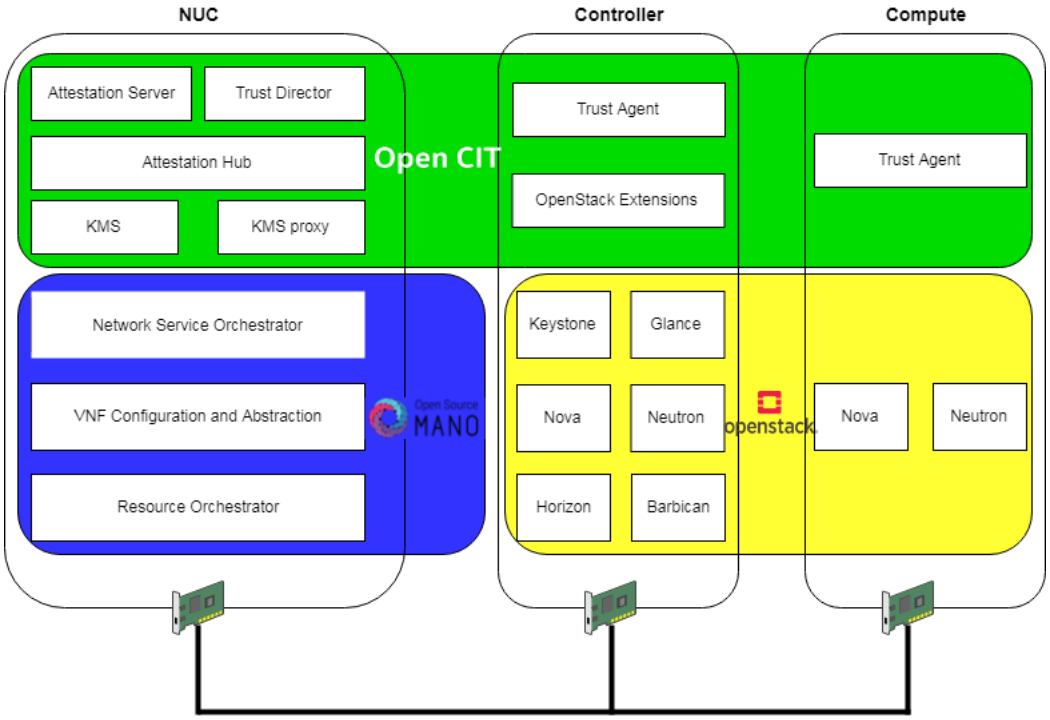

Figura 5.1. Soluzione dal punto di vista logico.

vista logico tutti i componenti coinvolti e la loro localizzazione nei server fisici utilizzati. Come è possibile vedere sono state utilizzate tre macchine differenti. Nelle macchine denominate Controller e Compute è presente il Trust Agent. Un vincolo fondamentale nella realizzazione di un server che possa essere verificato ed attestato è il suo avvio in modalità trusted boot cioè è necessario partire da un server fisico sicuro per estendere la catena di fiducia e verificare anche le macchine virtuali sviluppate su di esso. L'installazione del Trust Agent, mostrata nel manuale in Appendice A, prevede la necessità che esso venga installato su una macchina fisica con Intel TXT abilitato e con la possibilità di acquisire il possesso del TPM tramite l'operazione *take-ownership*. La scelta di avere il Trust Agent sul nodo Compute dà la possibilità di estendere la catena di fiducia. Dopo aver verificato l'attestazione del server fisico sarà possibile controllare anche l'integrità delle macchine virtuali che dovranno essere avviate su di esso. La presenza del Trust Agent anche nel nodo Controller permette di verificare che quel server fisico è fidato e non è stato manomesso. Nonostante sul Controller non dovranno essere avviate macchine virtuali, senza la presenza del Trust Agent un utente malevolo avrebbe potuto prenderne possesso e avrebbe potuto creare nuove istanze oppure avrebbe potuto modificare o eliminare quelle già esistenti creando dei danni agli utilizzatori del servizio.

Un altro componente inserito nel controller node sono le OpenStack Extensions necessarie all'interazione fra i software OpenStack ed Open CIT. Dopo aver installato le OpenStack extensions, la dashboard OpenStack permetterà la visualizzazione dei compute node utilizzati con il relativo stato di attestazione dell'hypervisor. Quando si cercherà di far partire una macchina virtuale sarà visibile lo stato di attestazione della macchina virtuale, la policy utilizzata e se l'avvio è stato regolare o se c'è stato un errore.

Relativamente ad OpenStack possiamo notare che i nodi Controller e Compute hanno dei componenti che sembrano replicati, in realtà le funzionalità principali sia di Nova sia di Neutron

sono gestite dal Controller mentre nel Compute è presente un agent attraverso cui è possibile pilotare il comportamento da adottare su esso e le funzionalità da abilitare.

L'altra macchina su cui sono stati creati dei componenti è un mini PC Intel denominato NUC. Questa macchina contiene le interfacce grafiche che permettono di impostare le caratteristiche di sicurezza in Open CIT. Il Trust Director consente di creare le Trust policy virtualizzate e non virtualizzate. Esso permette la scelta dei file e delle cartelle da verificare all'avvio del server fisico nel caso di policy non virtualizzata o quelli da controllare prima di lanciare la macchina virtuale nel caso di policy virtualizzata. La Trust policy non virtualizzata verrà caricata tramite SSH nel file system del server e si compone principalmente di due file che sono `trustpolicy.xml` e `manifest.xml`. Il file `manifest.xml` contiene le informazioni sull'algoritmo utilizzato per calcolare l'hash sui file e sulle cartelle, Open CIT 3.2 utilizza lo SHA256, inoltre è presente la lista di file e cartelle che devono essere misurati nella fase di avvio. La soluzione realizzata prevede l'inserimento dei file che appartengono all'installazione Open CIT in modo da rilevare eventuali modifiche su di essi e capire se l'host può essere considerato fidato. Saranno presenti i file del tbootxm, del tpm tools, del Trust Agent, delle OpenStack Extensions, del Policy Agent, del vRTM e verrà aggiunto un ulteriore file denominato `/trust-untrust` il cui contenuto verrà modificato per permettere la simulazione dei vari casi di test trattati nel Capitolo 7. Il file `trustpolicy.xml` è caratterizzato da informazioni relative all'immagine del sistema operativo che dovrà essere avviata ed il relativo hash. Sarà presente anche la policy utilizzata che nel caso di server fisico sarà measure only perché deve essere sempre possibile avviare la macchina anche se non è considerata fidata. Sarà presente l'algoritmo di hash utilizzato e la lista di misure di riferimento sia per i file sia per le cartelle scelti nella policy. Alla fine il file è firmato digitalmente dal Trust Director e dall'Attestation Server.

L'Attestation Server permette di registrare gli host fisici su cui dovrà essere eseguita l'attestazione. Esso si collegherà direttamente alla porta su cui è in ascolto il Trust Agent per richiedere le misure attuali calcolate. Durante la registrazione sarà possibile stabilire il tipo di misure da importare (BIOS MLE, VMM MLE) e sarà possibile specificare il meccanismo con cui creare una connessione sicura tramite TLS. Le misure importate verranno memorizzate nel database dell'Attestation Server e verranno utilizzate successivamente come valori di riferimento per l'attestazione. Questo server fornisce anche un'interfaccia grafica chiamata Trust dashboard da cui è immediato visualizzare gli host registrati e il risultato della loro attestazione. Il risultato è noto come Trust report e permetterà di visualizzare la lista di valori presenti nei vari PCR consentendo il confronto fra quelli memorizzati come riferimento e quelli attuali.

L'altro componente che espone un'interfaccia grafica per un utente che ha il compito di creare i servizi di rete ed istanziarli è il NSO di Open Source MANO. L'utente potrà, tramite un compositore, creare le VNF e i NS che verranno validati. NSO traduce automaticamente VNF ed NS nei corrispondenti descrittori ma la validazione serve a verificare se i valori inseriti dall'utente sono valori possibili per quei campi. NSO offre all'utente la possibilità di scegliere il servizio di rete da avviare e di accedere alle console delle macchine virtuali del servizio avviato.

Le funzionalità dei vari blocchi logici illustrati nella Figura 5.1 sono state descritte dettagliatamente nel Capitolo 4 ma per alcuni di essi è necessario impostare delle caratteristiche specifiche fin dalla fase di installazione al fine di svolgere il loro compito in maniera corretta all'interno dell'architettura realizzata.

I principali elementi che possono essere configurati in fase di installazione nell'Attestation Server sono:

- indirizzo ip e porta su cui è raggiungibile il server;
- tipologia del web server (attualmente l'unica tipologia supportata è Tomcat);
- database utilizzato (attualmente l'unica tipologia supportata è Postgres);
- configurazione delle credenziali per alcuni tipi di account come quello principale di amministrazione chiamato MC_FIRST, quello per accedere al web server Tomcat, quello per il funzionamento da Privacy CA.

Il funzionamento dell'attestazione remota avviene secondo il meccanismo descritto nel Capitolo 2 con attestazione tramite Privacy CA. L'Attestation Server in questa configurazione svolgerà il doppio compito di verificatore e di Privacy CA anche se nella fase di installazione si potrebbe indicare l'indirizzo ip di un Privacy CA esterno. L'interazione fra Trust Agent ed Attestation Server prevede che nella prima fase il Trust Agent, già in possesso del certificato del Privacy CA, generi ed ottenga la verifica delle sue AIK. Nella seconda fase sfrutterà le AIK per inviare all'Attestation Server le misure eseguite sui file indicati nella Trust policy al fine di verificare il loro contenuto e confrontarlo con quello di riferimento.

5.3 Integrazione software

Questa Sezione tratta le caratteristiche e le modifiche da apportare per integrare insieme i software in modo che collaborino per avviare un servizio di rete. L'avvio di un servizio di rete potrà avvenire con successo solo se il server e le macchine virtuali sono state attestate con risultato positivo.

L'integrazione fra Open Source MANO ed OpenStack è automatica. OSM supporta nativamente il VIM OpenStack quindi dopo aver installato OSM sarà necessario configurarlo all'utilizzo di OpenStack come VIM. Si dovrà accedere al RO in cui si indicherà che il data center da aggiungere è di tipo OpenStack. Si configurerà l'endpoint pubblico di keystone in modo che questo servizio possa essere autorizzato quando vuole accedere al VIM e si configureranno le credenziali per potervi accedere.

L'integrazione fra Open CIT ed OpenStack è automatica grazie alle OpenStack Extension create da Open CIT. L'installazione di queste estensioni implica la modifica di numerosi file per la corretta cooperazione fra i due software. Un requisito iniziale è che il Controller OpenStack abbia i servizi Nova ed Horizon installati ed in esecuzione. I principali cambiamenti apportati sono:

- Horizon: viene aggiunto un modulo che permetta di visualizzare nella dashboard le informazioni sullo stato dell'attestazione ed i geo tag (è il meccanismo che permette di avviare una macchina virtuale solo se rispetta dei vincoli di localizzazione definiti tramite delle etichette). Per le macchine virtuali viene aggiunto il tipo di policy utilizzata ed il suo stato.
- Nova: viene modificato il file di configurazione di Nova con una nuova sezione in cui vengono indicati i dati per utilizzare correttamente Open CIT. Sarà necessario indicare l'indirizzo IP dell'Attestation Server, le credenziali di accesso, gli URL per accedere alle attestazioni e alle altre API fornite, il percorso in cui trovare il certificato dell'Attestation Server per creare una connessione sicura.

L'integrazione fra Open CIT ed Open Source MANO non è prevista automaticamente quindi è stato necessario implementare dei meccanismi che permettano il loro utilizzo congiunto. L'esigenza principale è stata quella di recuperare, in fase di avvio di una nuova macchina virtuale, le attestazioni sia del server fisico su cui si dovrà sviluppare il servizio sia delle macchine virtuali che lo compongono. Queste operazioni sono state eseguite nel RO e prevedono:

1. creazione di una connessione sicura verso l'Attestation Server tramite HTTPS;
2. estrazione delle attestazioni associate ai server fisici;
3. esito della verifica di integrità eseguita sulla VM che è stata avviata.

Il recupero dei dati dall'Attestation Server è possibile grazie alle REST API esposte. L'utilizzo di una connessione sicura e delle credenziali valide permette di autenticarsi correttamente con l'Attestation Server e di ricevere i dati richiesti in un formato opportuno.

Le API utilizzate durante il lavoro di tesi sono descritte nel Capitolo 6 mentre una descrizione delle principali API esposte dall'Attestation Server è inserita nel Manuale dello sviluppatore in Appendice B.

5.4 Estensione Open Source MANO

La sezione precedente ci mostra come sia necessario implementare delle operazioni che permettano ad Open Source MANO di recuperare le informazioni sull'attestazione dei server fisici e delle macchine virtuali. In base ai compiti che ogni componente di OSM svolge, già descritti nel Capitolo 4, è possibile capire che il Resource Orchestrator verrà esteso perché è l'elemento che si occupa dell'interazione col VIM e che colleziona tutte le informazioni necessarie alla creazione della nuova istanza. RO è il componente derivato dal precedente progetto OpenMANO [53], la Figura 5.2 mostra nel dettaglio i componenti interni ad RO ed evidenzia l'elemento che dovrà essere esteso:

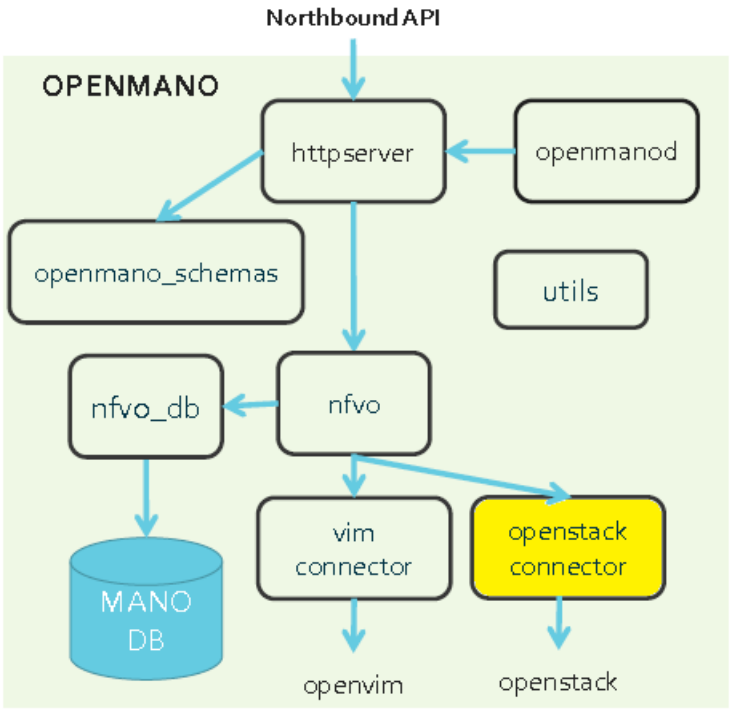

Figura 5.2. Componenti OpenMANO (fonte:[54]).

- Northbound API: è un'interfaccia attraverso cui i componenti esterni possono interagire col RO. Viene utilizzata sia dal NSO sia dal client OSM esterno. Tale client permette di estrarre informazioni o impartire dei comandi direttamente da linea di comando;
- openmanod: è il server openmano cioè il programma principale che implementa le funzionalità openmano. Esso carica le configurazioni da un apposito file contenente informazioni sulle porte utilizzate, sui database e sulle relative credenziali di accesso, inoltre avvia il programma httpserver con cui si mette in ascolto delle richieste che arrivano attraverso le API REST;
- httpserver: espone le API openmano con cui è possibile prelevare, modificare o aggiungere elementi come progetti o data center. Le operazioni richieste verranno inviate al componente nfvo che si occuperà di gestirle chiamando i metodi appropriati. Questo componente invia ad openmano_schemas i descrittori che riceve in formato YAML o JSON per un'opportuna verifica;
- openmano_schemas: è lo schema YAML o JSON utilizzato per validare le richieste ricevute dalle API REST;
- utils: è una libreria contenente le principali funzioni utilizzate dai vari componenti openmano;
- nfvo: è il motore principale di openmano. Implementa tutti i metodi per la creazione, cancellazione e gestione di VM, VNF ed NS. Interagisce con nfvo_db per prelevare o inserire

informazioni nel database ed interagisce col VIM per comunicare le operazioni da svolgere o per ricevere i risultati delle precedenti operazioni;

- nfvo_db: implementa tutti i metodi per interagire col database openmano. Riceve in ingresso i comandi dal componente nfvo e li traduce in appositi comandi per il database;
- connector: interagisce col VIM sottostante traducendo i comandi ricevuti dal nfvo in comandi per il VIM appropriato. I comandi possono essere legati alla gestione di utenti, progetti, servizi di rete, regole di sicurezza etc. La Figura 5.2 mostra solo i connettori OpenVIM ed OpenStack ma in realtà l'ultima versione di Open Source MANO offre il supporto anche ad Amazon Web Services e VMware.

Possiamo notare che il componente adatto ad essere esteso è l'OpenStack connector, al suo interno troveremo delle funzioni che convertono i comandi associati alla creazione di nuove istanze nei corrispondenti comandi OpenStack. Durante la creazione di un servizio di rete l'unità minima che è necessario allocare per realizzarlo è la macchina virtuale. La funzione che dovrà essere estesa aggiungendo le funzionalità di attestazione dei server fisici e delle VM è proprio quella che permette di creare nuove macchine virtuali.

5.5 Principi di funzionamento della soluzione

Identificato il file da modificare è stato necessario implementare un meccanismo che permetta di istanziare una nuova VM. Il server fisico su cui dovrà essere avviata la VM dovrà avere un'attestazione valida, inoltre dovrà essere verificato anche lo stato dell'integrità dell'immagine da avviare. La Figura 5.3 mostra il diagramma di flusso rappresentativo dell'algoritmo con cui è stata implementata la soluzione. Esso mette in rilievo le modifiche che sono state apportate.

Le operazioni che verranno eseguite sono:

1. viene richiesto l'avvio di una nuova istanza e vengono caricate le informazioni principali relative all'Attestation Server e agli host fisici presenti;
2. si stabilisce una connessione sicura con l'Attestation Server e ci si autentica;
3. si verifica la connessione e l'autenticazione. Se uno di questi processi dovesse dare problemi verranno inserite le informazioni in un apposito file di log e verrà generata un'eccezione che si propagherà a ritroso fino al NSO dove verrà gestita;
4. se la precedente verifica va a buon fine verrà richiesto all'Attestation Server il Trust report dei nodi fisici registrati e verrà analizzato per ricavare le informazioni sull'attestazione;
5. se uno dei nodi è non fidato verrà aggiornato il file di log e verrà generata un'eccezione che si propagherà a ritroso fino al NSO dove verrà gestita;
6. se i nodi sono fidati si inseriranno queste informazioni nel file di log e si potrà proseguire con la fase successiva;
7. si eseguiranno dei controlli sulle reti, sulle risorse da allocare, sull'immagine, sui metadati, sulla configurazione iniziale, sugli eventuali dischi da agganciare alla VM e poi si avvierà la macchina virtuale;
8. si verifica lo stato della VM e se risulta in stato di errore si aggiorna il log e si lancia un'eccezione che si propagherà a ritroso fino al NSO dove verrà gestita;
9. se lo stato è attivo quindi è stato possibile avviare la VM si eseguirà una nuova connessione verso l'Attestation Server;
10. si verifica la connessione e l'autenticazione. Se uno di questi processi dovesse dare problemi verranno inserite le informazioni in un apposito file di log e verrà generata un'eccezione che si propagherà a ritroso fino al NSO dove verrà gestita;

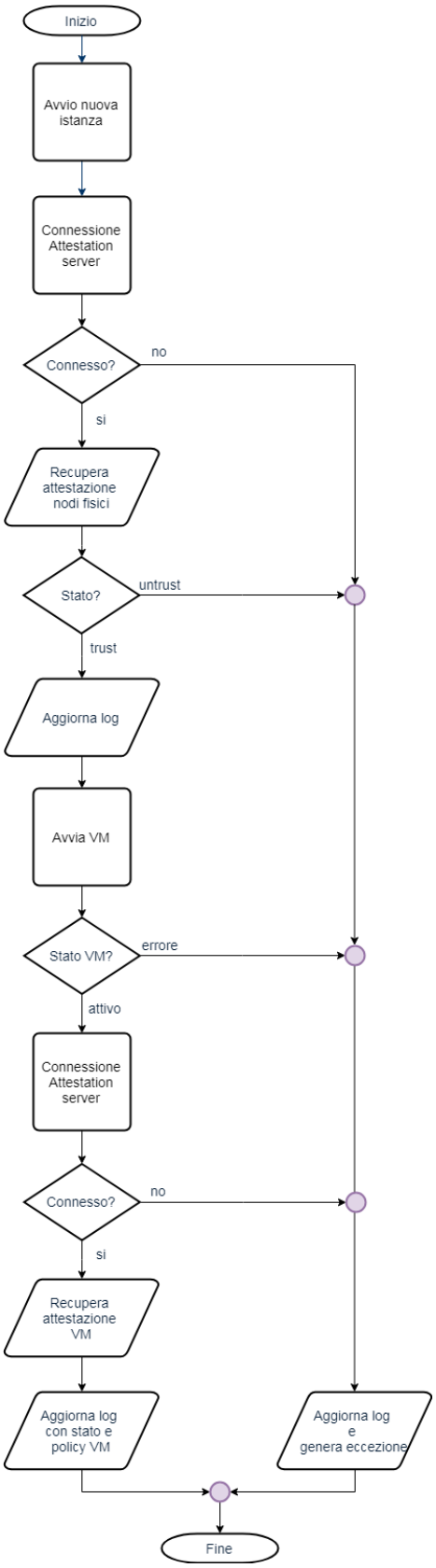

Figura 5.3. Diagramma di flusso della soluzione implementata.

11. se la precedente verifica va a buon fine si richiederà il Trust report associato alla VM e si estrarranno le informazioni sullo stato della VM e sulla sua policy;

12. si aggiornerà il log con le informazioni appena estratte.

Bisogna notare che è necessario collegarsi all'Attestation Server due volte. Nella prima fase si deve stabilire se il server fisico è adeguato ad ospitare la macchina virtuale. Nella seconda fase si deve attendere l'avvio della macchina virtuale per acquisire dall'Attestation Server il Trust report della VM. È possibile che venga avviata qualche macchina virtuale con stato non fidato. Ciò si verifica se la policy inserita è di tipo hash only, in questo caso nel file di log verrà inserito un messaggio di warning a causa dello stato di insicurezza in cui è stato sviluppato il servizio di rete ma sarà comunque permesso l'avvio del servizio. Nel Capitolo 6 verrà analizzato nel dettaglio il codice implementato.

5.6 Flusso di lavoro della soluzione

La soluzione realizzata prevede l'interazione di componenti appartenenti ad Open Source MANO ed Open CIT ed indirettamente vengono coinvolti anche i servizi OpenStack. La Figura 5.4 illustra

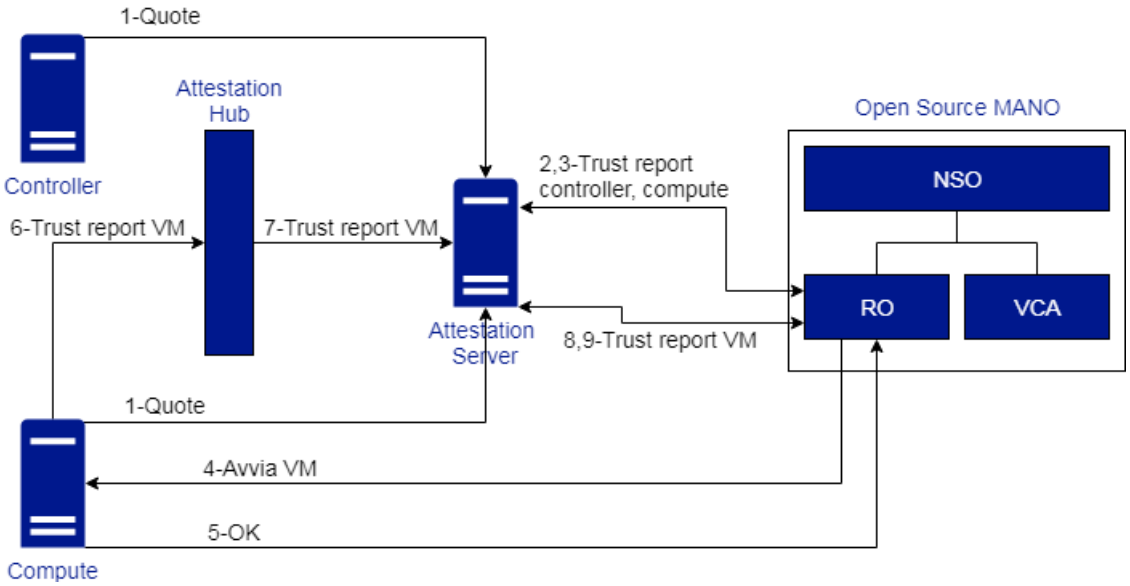

Figura 5.4. Workflow della soluzione realizzata.

gli elementi coinvolti quando è necessario avviare una macchina virtuale.

1. All'avvio i nodi Controller e Compute, sui quali sarà installato il Trust Agent, misurano i file indicati nella Trust policy ed inviano queste informazioni all'Attestation Server che sarà in grado di verificare se il loro stato è fidato;

2. Open Source MANO dovrà avviare una nuova macchina virtuale quindi richiede tramite il suo Resource Orchestrator il Trust report associato agli host fisici coinvolti in queste operazioni;

3. l'Attestation Server, dopo aver autenticato il RO, invia i Trust report richiesti;

4. nel caso in cui le verifiche effettuate da RO siano positive cioè sia Controller sia Compute sono fidati sarà possibile inviare al Compute la richiesta con cui avviare una macchina virtuale;

5. il Compute verifica la disponibilità delle risorse richieste. Se l'immagine da cui avviare la VM è cifrata richiede la chiave per decifrarla. Verifica che le misurazioni effettuate sui file dell'immagine corrispondano a quelle memorizzate nella Trust policy. Se tutte queste operazioni vanno a buon fine lancia la VM e conferma l'evento;

6. periodicamente l'Attestation Hub riceve i Trust report delle varie macchine virtuali avviate;
7. l'Attestation Hub invia all'Attestation Server i nuovi Trust report;
8. il RO richiede all'Attestation Server il Trust report associato alla macchina virtuale che è stata avviata;
9. l'Attestation Server invia il Trust report richiesto così il RO potrà estrarre i dati relativi allo stato di fiducia della macchina virtuale ed alla policy con cui è stata avviata.

La soluzione realizzata modifica il tipico flusso di lavoro dei software coinvolti. L'avvio di un servizio di rete che verifichi l'attestazione degli host fisici e delle macchine virtuali coinvolge sia Open Source MANO sia Open CIT. OpenStack non è direttamente influenzato dalla soluzione realizzata ma solo dalla cooperazione con Open CIT. Esso aggiunge dei criteri di selezione del nodo Compute basati sull'attestazione e modifica il servizio Glance per supportare l'inserimento di un archivio contenente l'immagine eventualmente cifrata e la Trust policy collegata ad essa.

Le modifiche al flusso di lavoro di Open Source MANO sono delle interazioni aggiuntive che permettono di acquisire i dati legati all'attestazione. La Figura 5.5 illustra le principali interazioni

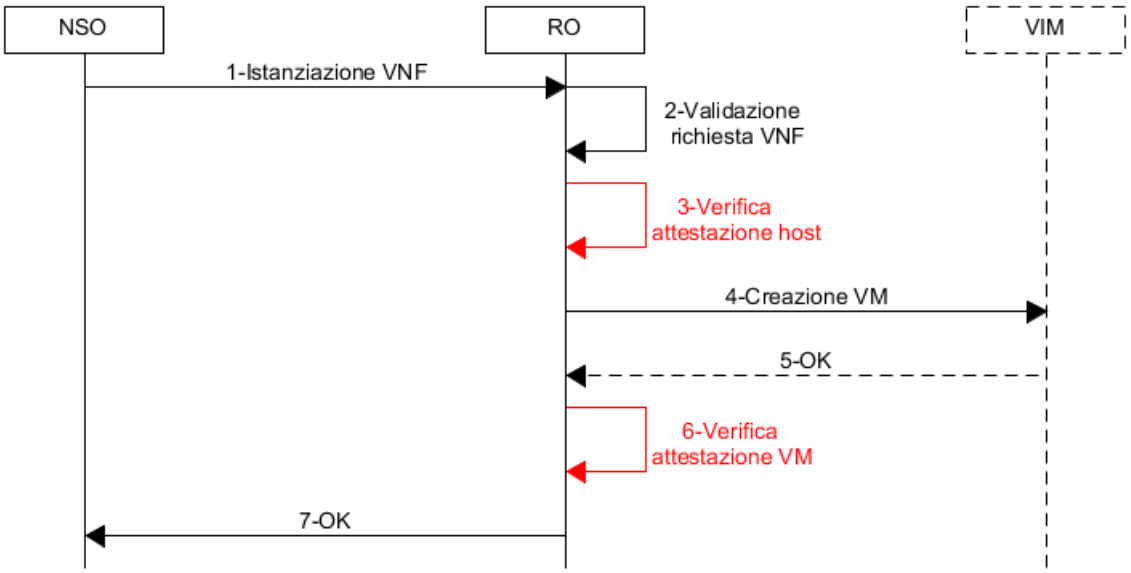

Figura 5.5. Modifiche del workflow Open Sorce MANO.

nel caso di avvio di una VNF ed evidenzia in rosso le interazioni introdotte con la soluzione realizzata. I componenti interni ad OSM e le operazioni generate da essi sono rappresentate con delle linee continue mentre il VIM e le sue operazioni sono rappresentate mediante linee tratteggiate per evidenziare che è un componente esterno ad OSM. Il flusso di lavoro prevede:

1. NSO invia il descrittore del VNF al RO;
2. RO valida la richiesta ricevuta eseguendo sia controlli sintattici cioè se il descrittore è strutturato bene sia controlli semantici cioè se i valori associati ai vari campi sono fra quelli ritenuti validi;
3. RO verifica l'attestazione degli host fisici. Se qualcuno di essi è in stato non fidato verrà lanciata un'eccezione che sarà gestita dal NSO altrimenti è possibile proseguire l'esecuzione;
4. RO possiede i dati per raggiungere il VIM ed invia ad esso le informazioni per realizzare la VM;

5. il VIM verifica la disponibilità delle risorse richieste, in cooperazione con Open CIT controlla l'attestazione e la policy legata alla VM e se tutto va a buon fine comunica al RO che la macchina virtuale è stata creata correttamente. Un caso particolare è quando la policy è hash only e lo stato è non fidato infatti la comunicazione che arriverà ad RO indicherà solamente che è stato possibile avviare la VM. In caso di problemi il VIM solleverà un'eccezione;
6. RO acquisisce le informazioni sullo stato dell'attestazione e sulla policy con cui è stata avviata la macchina virtuale e li inserisce nell'apposito file di log;
7. RO invia ad NSO la conferma che è stato possibile creare il VNF .

Le operazioni illustrate fra i punti 3 e 6 si ripeteranno tante volte quante sono le VM che costituiscono il VNF. L'avvio di un NS prevederà l'avvio delle singole VNF di cui esso è costituito.

Il tipico flusso di lavoro di Open CIT viene modificato introducendo alcune interazioni che permettono di fornire i risultati legati all'attestazione degli host fisici e delle macchine virtuali. La Figura 5.6 illustra le principali interazioni nel caso di avvio di una macchina virtuale ed evidenzia in rosso le modifiche introdotte dalla soluzione realizzata. Il flusso di lavoro mostrato presuppone che sia già stato caricato su Glance l'archivio contenente l'immagine cifrata e la Trust policy della macchina virtuale:

1. L'Attestation Server recupera i Trust report associati agli host fisici e li comunica al RO che deciderà se è possibile proseguire con l'istanziazione della macchina virtuale;
2. il nodo Compute preleva l'immagine da Glance e verifica che sia cifrata. Il Trust Agent installato nel Compute richiede la chiave per decifrare l'immagine al KMS proxy;
3. il KMS proxy richiede all'Attestation Server l'attestazione del nodo Compute a cui dovrebbe essere inviata la chiave;
4. l'Attestation Server si rivolge direttamente al Compute interessato chiedendo i valori legati alla sua attestazione. Il funzionamento attuale di Open CIT prevede che il nodo Compute calcoli le misurazioni associate ai file appartenenti alla Trust policy al suo avvio e memorizzi tali valori. Ogni richiesta successiva di attestazione riceverà come risposta sempre le stesse misure calcolate all'avvio;
5. il Trust Agent comunica all'Attestation Server le misurazioni eseguite sui file e le cartelle indicati nella Trust policy;
6. l'Attestation Server invia al KMS proxy il SAML in cui sono state inserite le informazioni di attestazione del nodo Compute;
7. il KMS proxy invia al KMS l'identificativo della chiave che vorrebbe ricevere e il SAML in cui si evidenzia che quel Compute ha un'attestazione valida;
8. il KMS recupera la chiave di decifratura e la invia al KMS proxy;
9. il KMS proxy invia al Trust Agent la chiave di decifratura;
10. il Trust Agent decifra l'immagine. Il componente vRTM si occupa di misurare l'immagine e di eseguire il confronto con i valori di riferimento della Trust policy e se la verifica dell'integrità avviene con successo lancia la nuova istanza;
11. il Trust report generato sarà caricato su OpenStack. Periodicamente l'Attestation Hub acquisisce queste informazioni da OpenStack e le invia all'Attestation Server;
12. l'Attestation Server recupera il Trust report associato alla macchina virtuale appena avviata e lo fornisce ai processi che lo richiedono.

Bisogna notare che solamente i nodi che posseggono un'attestazione valida potranno ricevere la chiave per decifrare l'immagine e quindi potranno avviare una nuova istanza. Nel caso in cui l'attestazione del nodo fisico non fosse valida verrebbe sollevata un'eccezione che impedirebbe la prosecuzione del normale flusso di lavoro.

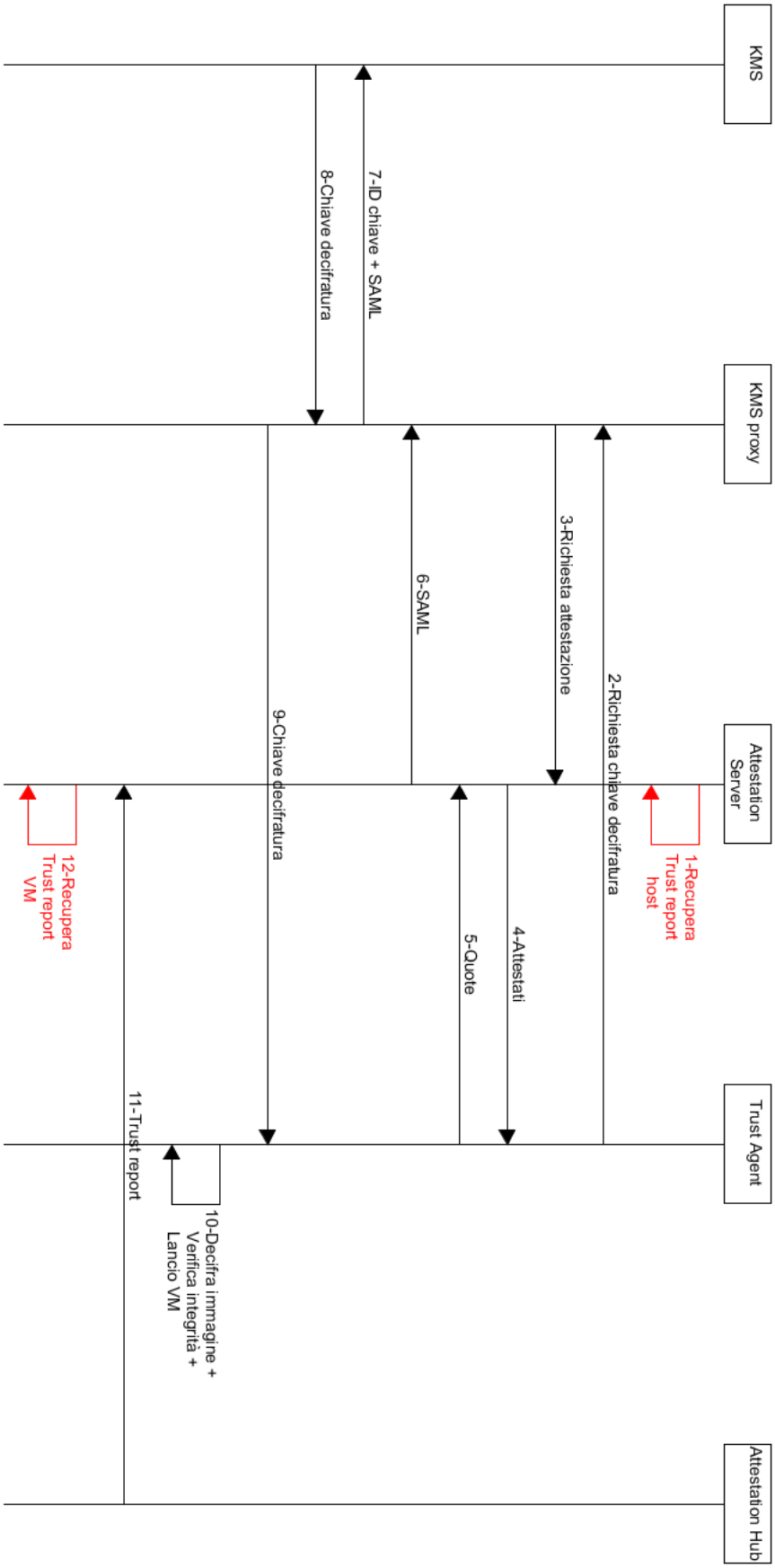

Figura 5.6. Modifiche del workflow Open CIT.

Capitolo 6

Implementazione

In questo Capitolo tratteremo l'implementazione della soluzione esposta nel Capitolo 5. La Sezione 6.1 analizzerà le principali API con cui è possibile accedere all'Attestation Server e le modifiche apportate al codice sorgente per svolgere queste operazioni. La Sezione 6.2 mostrerà la struttura del Trust report ed il meccanismo implementato per analizzarlo ed estrarre i dati di interesse. La Sezione 6.3 illustrerà come creare una connessione sicura verso l'Attestation server.

6.1 Attestation server API

I test eseguiti e le modifiche apportate durante il lavoro di tesi hanno previsto l'utilizzo di Open Source MANO TWO. In questa versione il file da modificare è localizzato all'interno del Resource Orchestrator nella cartella `/usr/lib/python2.7/dist-packages/osm_ro/`, siccome il VIM utilizzato è OpenStack il file python da modificare è `vimconn_openstack.py`. Le modifiche apportate sono, in parte, un riadattamento delle OpenStack extensions realizzate nel progetto Open CIT [42].

La prima analisi da effettuare è relativa alle API che permettono di contattare il server per estrarre le informazioni sull'attestazione. Un API che possiamo analizzare è la `searchHostAttestations` con cui è possibile richiedere all'Attestation Server le informazioni relative all'attestazione di un host fisico, essa è caratterizzata da:

- URL: `https://SERVER:PORTA/mtwilson/v2/host-attestations?nameEqualTo=HOST` indica come deve essere strutturata la richiesta per poter recuperare le attestazioni:
  - SERVER indica l'indirizzo IP o l'hostname associato all'Attestation Server.
  - PORTA indica la porta su cui è in ascolto il server e la configurazione di default prevede l'utilizzo della porta 8443.
  - HOST indica l'indirizzo IP o l'hostname associato all'host fisico di cui si stanno chiedendo le informazioni sull'attestazione. L'Attestation Server memorizza casualmente l'IP o l'hostname quindi sarà necessario richiedere i dati con entrambi gli elementi prima di constatare l'assenza di attestazioni per quell'host.
- header: contiene le informazioni necessarie all'autenticazione ed il formato del file contenente l'attestazione accettato dall'host. Il valore utilizzato è `application/samlassertion+xml` che prevede la restituzione del Trust report in formato SAML ed inoltre con questa modalità verrà restituita solamente l'ultima attestazione valida che coincide con i criteri di ricerca specificati. Altri possibili valori sono `application/xml` e `application/json` che restituiscono l'intera lista di attestazioni associate a quei criteri nei formati XML o JSON;
- parametri: è possibile raffinare la ricerca specificando più parametri concatenati attraverso il simbolo &. Alcuni parametri possibili sono:
  - fromDate: per indicare la data di partenza da cui cominciare a cercare;

  - toDate: per specificare la data oltre la quale non cercare più nulla;
  - limit: per indicare un numero massimo di attestazioni che possono essere restituite;
  - numberOfDays: per specificare il numero di giorni precedenti a quello attuale da cui si dovranno cominciare a cercare le attestazioni.

Un'altra API utilizzata durante la tesi è la `searchVMAttestations` che permette di richiedere all'Attestation Server le informazioni relative all'attestazione di una macchina virtuale, essa è caratterizzata da:

- URL: `https://SERVER:PORTA//mtwilson/v2/vm-attestations?vmInstanceIdEqualTo=VM_ID` indica come deve essere strutturata la richiesta per poter richiedere le attestazioni:
  - SERVER indica l'indirizzo IP o l'hostname associato all'Attestation Server.
  - PORTA indica la porta su cui è in ascolto il server e la configurazione di default prevede l'utilizzo della porta 8443.
  - VM_ID indica l'identificativo della macchina virtuale di cui si stanno chiedendo le informazioni sull'attestazione.

  Sarebbe possibile effettuare una ricerca anche per host tramite il parametro hostNameEqualTo ma verrebbero restituite tutte le attestazioni associate alle VM che sono state avviate su quell'host;
- header: contiene le informazioni necessarie all'autenticazione e il formato del file contenente l'attestazione accettato dall'host. Il valore utilizzato è `application/xml` che prevede la restituzione del Trust report in formato XML cioè verrà restituito il SAML con i dati di attestazione e numerose altre informazioni. Un altro possibile valore è `application/json` che restituisce l'attestazione associata a quei criteri nel formato JSON;
- parametri: è possibile raffinare la ricerca specificando più parametri concatenati attraverso il simbolo &. Alcuni parametri possibili sono:
  - fromDate: per indicare la data di partenza da cui cominciare a cercare;
  - toDate: per specificare la data oltre la quale non cercare più nulla;
  - limit: per indicare un numero massimo di attestazioni che possono essere restituite;
  - numberOfDays: per specificare il numero di giorni precedenti a quello attuale da cui si dovranno cominciare a cercare le attestazioni.

La Figura 6.1 illustra la parte di codice contenente le variabili ed i valori che è stato necessario impostare per ottenere il comportamento desiderato. La prima linea indica il massimo numero di secondi che l'Attestation Hub impiega per verificare su OpenStack se sono presenti nuove attestazioni associate alle macchine virtuali e per caricarle sull'Attestation Server. La seconda linea indica il numero massimo di secondi che è necessario attendere prima di considerare un server irraggiungibile. A seguire sono inserite le informazioni relative ad hostname ed indirizzo IP di ogni host fisico su cui dovrà essere verificata l'attestazione prima di avviare qualsiasi macchina virtuale. Le successive linee mostrano i dati necessari per comunicare con l'Attestation Server, in particolare sono indicati:

- l'indirizzo IP a cui è raggiungibile;
- la porta su cui è in ascolto;
- le credenziali attraverso cui è possibile autenticarsi espresse nel formato `USERNAME:PASSWORD`. Sono necessarie credenziali distinte per richiedere l'attestazione degli host fisici e l'attestazione delle macchine virtuali;
- gli indirizzi delle API che si dovranno utilizzare per estrarre le informazioni sull'attestazione degli host fisici e delle macchine virtuali.

```
attestation_hub_refresh = 120
server_timeout = 180

REGISTERED_HOSTS = {
    'controller1': {
        'hostname': 'controller',
        'ip': '130.192.1.90'
    },
    'compute1': {
        'hostname': 'compute1',
        'ip': '130.192.1.91'
    }

ATTESTATION_SERVICE = {
    'IP': '130.192.1.110',
    'port': '8443',
    'auth_blob': 'tagadmin:PASS_TAGADMIN',
    'auth_blob_vm': 'admin:PASS_ADMIN',
    'search_host_attestations': '/mtwilson/v2/host-attestations?nameEqualTo=',
    'search_vm_attestations': '/mtwilson/v2/vm-attestations?vmInstanceIdEqualTo
        ='
}

}
```

Figura 6.1. Variabili inserite nel file `vimconn_openstack.py`.

6.2 Parsing del Trust report

Le informazioni fornite dall'Attestation Server sono inserite all'interno di un apposito file denominato Trust report. Esso viene generato sia per le attestazioni degli host fisici sia per le attestazioni delle macchine virtuali. Nel caso di informazioni sull'attestazione di un host fisico sarà possibile richiedere solo una parte del Trust report denominata SAML assertion da cui è possibile estrarre le informazioni principali relative all'attestazione, in particolare saranno presenti:

- l'emettitore e l'istante di emissione della SAML assertion;
- l'host fisico a cui si riferisce;
- lo stato di fiducia complessivo della piattaforma;
- lo stato di fiducia collegato alle BIOS MLE ed alle VMM MLE.

La soluzione implementata permette di eseguire il parsing della SAML assertion utilizzando una funzione che analizza i dati ed estrae lo stato di fiducia. I passi eseguiti dalla funzione prevedono:

1. la conversione dei dati in una stringa per essere facilmente manipolati dalla funzione;
2. la ricerca della lista di elementi presenti;
3. per ogni elemento si cerca quello che possiede un attributo `Name` con valore `trusted`;
4. si estrae il testo corrispondente all'attributo specificato nel punto precedente.
5. la funzione restituisce il valore estratto.

La soluzione permette di gestire anche i casi in cui i dati passati a questa funzione non sono nel formato corretto ed in tale situazione viene aggiornato il log e viene generata un'eccezione.

Nel caso di informazioni sull'attestazione di una macchina virtuale si riceverà il Trust report in formato XML da cui sarà possibile estrarre le informazioni su:

- il nome dell'host su cui è stata avviata la macchina virtuale;
- l'identificativo della macchina virtuale a cui appartiene il Trust report;
- la SAML assertion associata a questa macchina virtuale che conterrà informazioni sullo stato di integrità e la policy della VM;
- l'algoritmo di hash utilizzato per calcolare le misure;
- la lista delle misure di riferimento definite nella Trust policy;
- la lista di misure attuali calcolate prima dell'avvio della macchina virtuale.

La soluzione implementata estrarrà la parte relativa alla SAML assertion e la passerà ad una funzione che analizza i dati ed estrae sia lo stato di fiducia sia la policy utilizzata per avviare la macchina virtuale. I passi eseguiti dalla funzione prevedono:

1. la conversione dei dati in una stringa per essere facilmente manipolati dalla funzione;
2. la ricerca della lista di elementi presenti;
3. per ogni elemento si cerca quello che possiede un attributo `Name` con valore `vm_trust_status`;
4. si estrae il testo corrispondente all'attributo specificato nel punto precedente.
5. per ogni elemento si cerca quello che possiede un attributo `Name` con valore `vm_trust_policy`;
6. si estrae il testo corrispondente all'attributo specificato nel punto precedente.
7. la funzione restituisce i valori estratti.

La soluzione permette di gestire anche i casi in cui i dati passati a questa funzione non sono nel formato corretto ed in tale situazione viene aggiornato il log e viene generata un'eccezione.

Tutte le informazioni relative allo stato degli host fisici e delle macchine virtuali o relative alle eccezioni vengono inserite in un file di log che in Open Source MANO TWO è il file `openmano.log` localizzato all'interno del Resource Orchestrator nella cartella `/var/log/osm/`. L'operatore che gestisce i servizi di rete potrà accedere al file di log per conoscere le principali informazioni associate ai servizi di rete sviluppati, potrà conoscere:

- lo stato di fiducia degli host fisici coinvolti;
- lo stato di fiducia delle macchine virtuali che costituiscono il servizio di rete;
- la policy associata ad ogni singola VM.

Inoltre visualizzerà un messaggio di warning quando lo stato delle macchine virtuali risulta non fidato ma il servizio verrà avviato ugualmente perché la policy è di tipo hash only.

6.3 Connessione all'Attestation Server

La necessità principale nella realizzazione di un meccanismo che permetta di interagire con Open CIT è quella di creare una connessione sicura verso l'Attestation Server attraverso cui è possibile ricavare le informazioni di interesse. I passi che è stato necessario seguire sono:

1. l'importazione del certificato dell'Attestation Server necessario a creare una connessione sicura. Si preleverà il certificato `ssl.crt` dalla cartella `/etc/intel/cloudsecurity` e si copierà all'interno del Resource Orchestrator;
2. nel file `vimconn_openstack.py` verranno importate le librerie necessarie a gestire la connessione HTTPS;
3. si instaura la connessione indicando l'indirizzo IP e la porta su cui è in ascolto l'Attestation Server. Per realizzare una connessione sicura sarà necessario indicare esplicitamente la posizione in cui è stato copiato il certificato dell'Attestation Server;
4. si crea l'header con cui dovrà essere effettuata la richiesta. In esso si inseriranno il nome utente e la password con cui eseguire l'autenticazione ed il formato accettato per le risposte. Sarà necessario utilizzare credenziali differenti per acquisire il Trust report degli host fisici e quello delle macchine virtuali. Il formato della risposta prevederà per gli host fisici la SAML assertion mentre per le macchine virtuali il formato XML;
5. si verificherà se è stata ricevuta correttamente la risposta. Se c'è stato un errore si farà un nuovo tentativo cambiando i dati utilizzati per identificare l'host di cui si vuole il report perché l'Attestation Server memorizza casualmente l'host in base all'indirizzo IP o all'hostname. Se nella prima richiesta si era provato l'IP allora il secondo tentativo si effettuerà con l'hostname. Se in entrambi i casi viene restituito un codice di errore allora verrà generata un'eccezione altrimenti sarà possibile analizzare il file ricevuto. La richiesta di attestazione per la macchina virtuale conterrà direttamente l'identificativo della VM quindi non sarà necessario contattare il server due volte per avere le informazioni richieste;
6. nel caso di stato non fidato di un host fisico si inseriranno le informazioni nel file di log e si lancerà un'eccezione che blocchi l'esecuzione attuale, liberi le risorse che erano state riservate e comunichi al NSO che non è stato possibile istanziare quella macchina virtuale.

La verifica che un host fisico è fidato permette di proseguire l'esecuzione avviando una macchina virtuale. Il programma implementato dovrà attendere uno stato consistente della macchina virtuale prima di estrarre le informazioni relative al suo stato di integrità. Gli stati che potrà assumere una macchina virtuale sono:

- ERROR: nel caso in cui non è stato possibile avviare la macchina virtuale. Di questa categoria fanno parte anche i casi in cui la policy della macchina virtuale è hash and enforce e il suo stato è non fidato. La verifica dell'integrità viene eseguita prima di lanciare l'istanza ed in questo caso l'istanza non verrà proprio avviata;
- ACTIVE: nel caso in cui è stato possibile avviare la macchina virtuale correttamente. Sarà possibile contattare l'Attestation Server per ricevere il Trust report ed estrarre lo stato della VM e la policy utilizzata. La verifica di integrità per le VM viene eseguita dal componente vRTM del Trust Agent e l'Attestation Hub inserirà periodicamente i nuovi risultati nell'Attestation Server;
- TIMEOUT: è il caso in cui scade il tempo di attesa per conoscere lo stato della VM. Ciò accade quando la macchina virtuale non è stata in grado di allocare tutte le risorse per avviarsi quindi viene lanciata un'eccezione che rilascerà le risorse occupate.

Capitolo 7

Testing

In questo Capitolo illustreremo i dettagli specifici con cui è stata implementata la soluzione nella Sezione 7.1. Mostreremo i descrittori che sono stati utilizzati per realizzare i test nella Sezione 7.2. Descriveremo l'interfaccia grafica degli elementi coinvolti nella Sezione 7.3. Discuteremo i test effettuati nelle Sezioni 7.4, 7.5, 7.6, 7.7, 7.8, 7.9. Infine commenteremo i test effettuati nella Sezione 7.10.

7.1 Dettagli sull'architettura realizzata

La configurazione utilizzata in laboratorio per lo svolgimento dei test prevede l'utilizzo di tre elaboratori distinti. Un elaboratore è un Intel NUC con CPU Intel core i5 di quinta generazione, 16 GB di RAM e sistema operativo Ubuntu 16.04.2 LTS. Gli altri due elaboratori sono due notebook HP entrambi con le stesse caratteristiche cioè CPU Intel i7 di terza generazione, 8 GB di RAM e sistema operativo Ubuntu 16.04.2 LTS.

La configurazione di rete utilizzata è mostrata nella Figura 7.1.

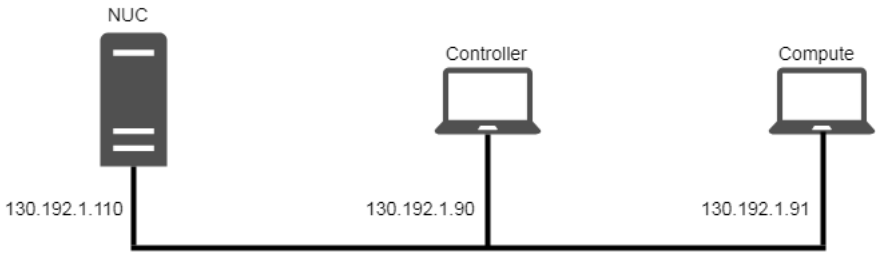

Figura 7.1. Soluzione con indirizzi di rete.

Il NUC conterrà l'installazione di Open Source MANO TWO e i principali servizi di Open CIT 3.2 cioè Attestation Server, Trust Director, KMS, KMS proxy ed Attestation Hub. I due notebook implementeranno i nodi Controller e Compute di OpenStack Mitaka, inoltre in entrambi sarà presente il Trust Agent Open CIT.

7.2 Descrittori utilizzati

I test realizzati prevedono lo sviluppo di un servizio di rete ed ottengono dei messaggi in cui viene esplicitato se è stato possibile avviare il servizio o il motivo per cui non lo è stato. La struttura di base del servizio di rete utilizzato in ogni test corrisponde a quello fornito in Open Source MANO

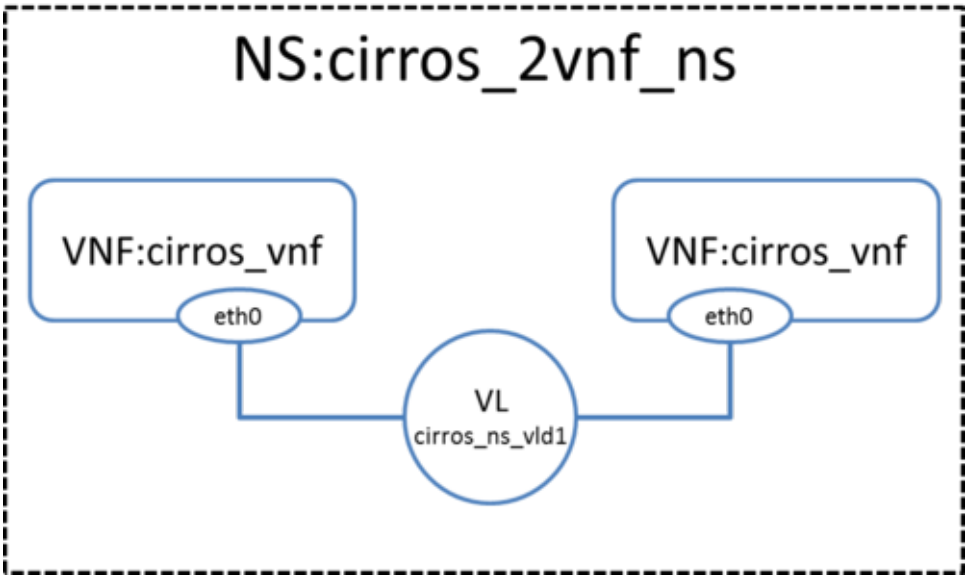

Figura 7.2. Servizio di rete Open Source MANO (fonte:[35]).

come servizio di rete di prova. Esso sarà costituito da due VNF ed un collegamento virtuale che le interconnette. La Figura 7.2. mostra come è strutturato il servizio di rete.

Ogni VNF sarà caratterizzata da un'unica macchina virtuale con:

- sistema operativo Cirros 0.3.4;
- una CPU virtuale;
- 512 MB di RAM;
- 1 GB di disco di memorizzazione.

Nella realizzazione dei test è stato necessario:

1. accedere al Trust Director per caricare la Trust policy non virtualizzata sui nodi Controller e Compute;
2. accedere al Trust Director per inserire un'immagine e la sua Trust policy virtualizzata su OpenStack Glance;
3. modificare il descrittore della VNF indicando di utilizzare come immagine quella appena caricata su Glance;
4. modificare il descrittore del servizio di rete specificando l'identificativo della nuova VNF da utilizzare.

7.3 Interfacce grafiche

L'utente che deve avviare un servizio di rete interagisce con l'interfaccia grafica esposta da Open Source MANO chiamata Launchpad. In essa potrà scegliere il servizio da avviare e potrà verificare se il servizio è stato avviato correttamente. Il file di log presente all'interno del Resource Orchestrator permetterà di visualizzare le informazioni dettagliate sullo stato di fiducia dei nodi Controller e Compute, sullo stato di integrità delle macchine virtuali e sulle policy utilizzate. Queste ultime informazioni saranno visualizzabili anche nella Trust Dashboard dell'Attestation Server

ed accedendo alla Dashboard di OpenStack. Una breve descrizione di come sono strutturate le interfacce grafiche ci permetterà di mostrare i risultati dei test in maniera agevole e facilmente comprensibile.

Il Launchpad conterrà le principali informazioni collegate al servizio di rete che si sta cercando di avviare:

- nome del servizio di rete;
- il nome presente nel catalogo OSM del descrittore del servizio di rete;
- lo stato attuale che può passare dalla fase di realizzazione a quella attiva o a quella di errore;
- il tempo trascorso dall'avvio del servizio;
- nel caso di servizio in stato attivo sarà possibile accedere alle console delle macchine virtuali che sono state avviate;
- l'icona con cui terminare il servizio di rete.

La Trust Dashboard conterrà le informazioni sui server fisici registrati presso l'Attestation Server e sul loro stato di fiducia, in particolare saranno presenti:

- i nomi degli host registrati;
- lo stato collegato agli asset tag;
- lo stato di fiducia legato alle BIOS MLE;
- lo stato di fiducia legato alle VMM MLE;
- lo stato di fiducia complessivo per la piattaforma;
- l'istante a cui si riferiscono questi dati;
- la Trust assertion cioè i dati dell'attestazione in formato SAML;
- il Trust report cioè la lista di valori attuali inseriti nei PCR confrontati con i valori di riferimento. Nel caso di stato non fidato verranno evidenziati i file e le cartelle che hanno valori differenti rispetto a quelli di riferimento;
- lo stato del collegamento fra l'Attestation Server e il Trust Agent.

La Dashboard OpenStack conterrà le informazioni legate alle macchine virtuali che sono state avviate per realizzare il servizio di rete richiesto. Essa è caratterizzata da:

- il nome dell'istanza;
- l'immagine utilizzata per avviare l'istanza;
- lo stato di attestazione dell'istanza che visualizza se la macchina virtuale è stata avviata con uno stato fidato, se gli asset tag sono presenti ed eventualmente il loro stato, la policy utilizzata per avviare la macchina virtuale;
- l'indirizzo IP assegnato all'istanza;
- la configurazione legata all'istanza (CPU, RAM, disco);
- lo stato attuale dell'istanza e se è attivo verrà indicata anche la fase attuale cioè se la macchina virtuale si sta avviando o se è già in esecuzione;
- il tempo trascorso da quando è stata avviata la macchina virtuale.

7.4 Test VM enforce trusted

Questo test simula l'avvio del servizio di rete denominato `nsd-enforce-trusted`. L'immagine caricata su Glance è la `enforce-encrypt-trusted` ed è associata ad una Trust policy di tipo hash and enforce che prevede la misurazione della cartella `/boot` contenente il kernel Linux ed i principali file necessari all'avvio del sistema.

La Figura 7.3 mostra i risultati ottenuti. Sul Launchpad è possibile notare che il servizio di

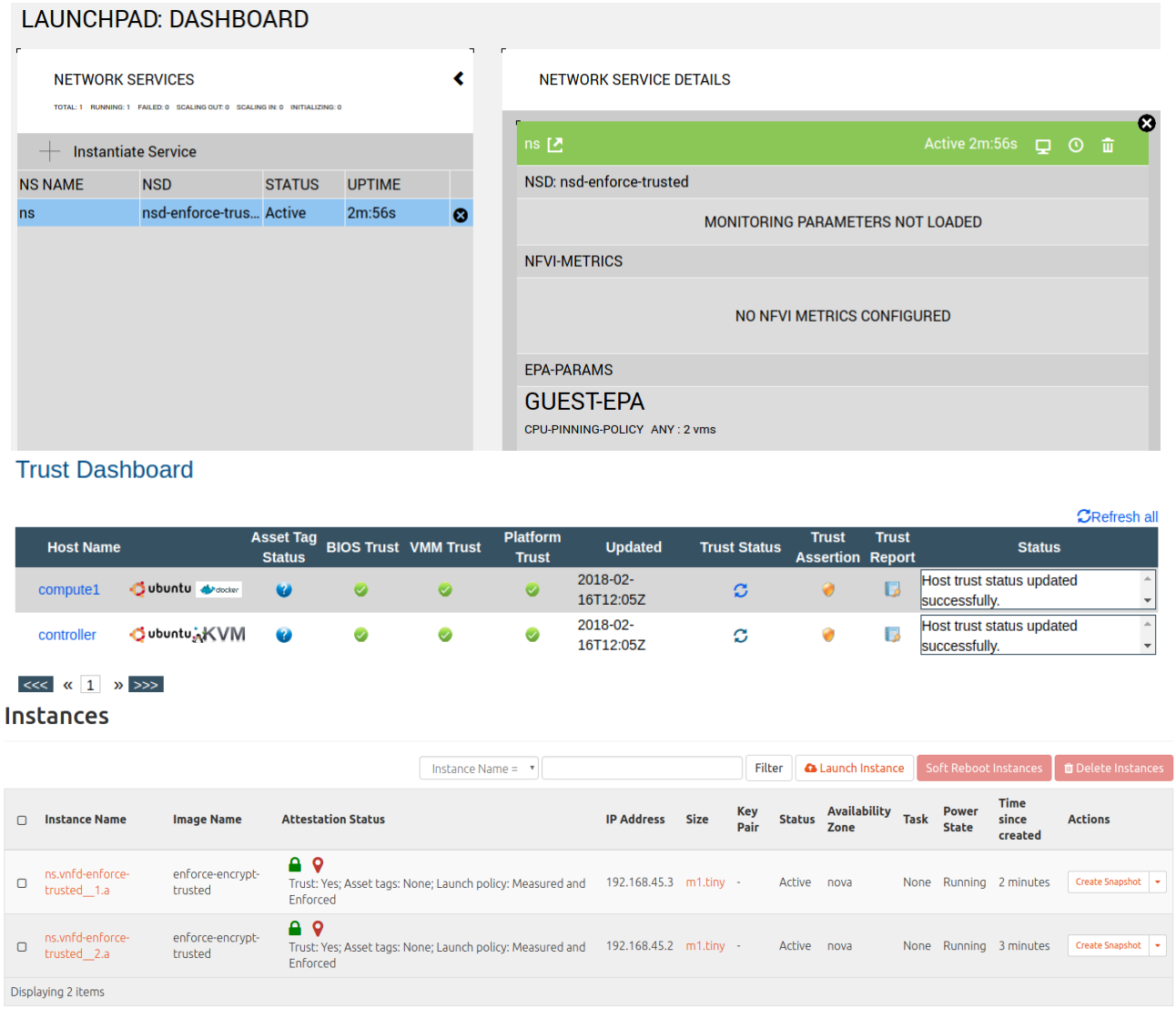

Figura 7.3. Risultati del Test VM enforce trusted.

rete `nsd-enforce-trusted` è stato avviato correttamente infatti il suo stato è attivo. La Trust Dashboard mostra che sia il nodo Controller sia il nodo Compute hanno un'attestazione valida. L'OpenStack Dashboard elenca le istanze avviate e per ognuna di esse visualizza lo stato che è fidato e la policy con cui sono state lanciate che è di tipo measured and enforced inoltre assegna ad ognuna di esse un indirizzo IP privato attraverso cui possono comunicare fra loro.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.4 permettono di vedere le stesse informazioni mostrate nelle interfacce grafiche. Si può notare come prima dell'avvio di ogni macchina virtuale venga verificata l'attestazione dei server fisici registrati.

```
2018-02-16T12:06:44 DEBUG openmano.vim.openstack vimconn_openstack.py:955 Host
     compute1(130.192.1.91) trust
2018-02-16T12:06:44 DEBUG openmano.vim.openstack vimconn_openstack.py:955 Host
     controller(130.192.1.90) trust
...
2018-02-16T12:07:41 DEBUG openmano.vim.openstack vimconn_openstack.py:1292 VM
    f4a3a4fc-23db-4e56-8165-e2a36f031d3f trust and run with policy
    MeasureAndEnforce!
...
2018-02-16T12:07:42 DEBUG openmano.vim.openstack vimconn_openstack.py:955 Host
     compute1(130.192.1.91) trust
2018-02-16T12:07:42 DEBUG openmano.vim.openstack vimconn_openstack.py:955 Host
     controller(130.192.1.90) trust
...
2018-02-16T12:08:43 DEBUG openmano.vim.openstack vimconn_openstack.py:1292 VM
    7218db5c-6bb4-4454-9bbc-87a7de17baae trust and run with policy
    MeasureAndEnforce!
```

Figura 7.4. Informazioni relative al Test VM enforce trusted presenti in `openmano.log`.

7.5 Test VM enforce untrusted

Questo test simula l'avvio del servizio di rete denominato `nsd-enforce-untrusted`. L'immagine caricata su Glance è la `enforce-encrypt-untrusted` ed è associata ad una Trust policy di tipo hash and enforce che prevede la misurazione della cartella `/bin` contenente i programmi di sistema essenziali.

L'immagine utilizzata ai fini del test è Cirros, una distribuzione minimale Linux che implementa le funzionalità dei principali programmi di sistema affidandosi a BusyBox. BusyBox [55] è un file eseguibile utilizzato nei sistemi limitati grazie alle sue dimensioni ridotte e alla sua velocità di risposta. Esso contiene al suo interno il codice binario per eseguire i principali comandi Linux e può essere utilizzato creando dei collegamenti col nome dei tipici comandi Linux che puntano all'eseguibile BusyBox. Cirros all'interno della cartella `/bin` non conterrà i normali binari che implementano i vari comandi ma avrà dei collegamenti a BusyBox attraverso cui sarà possibile eseguire tali comandi. Il sistema creerà questi collegamenti all'avvio per permettere il normale funzionamento delle applicazioni.

Il Trust Director attraverso l'operazione di montaggio dell'immagine misura i file e crea la Trust policy che conterrà i valori di riferimento. Il Trust Agent copia i file dell'immagine e prima di avviare l'istanza verifica che le misurazioni eseguite sui file corrispondano a quelle di riferimento. Questi differenti meccanismi implicano che la cartella `/bin` conterrà dei file diversi fra la creazione della Trust policy e la sua successiva verifica cioè durante la verifica non saranno ancora presenti i collegamenti a BusyBox quindi il risultato della verifica sarà non fidato.

La Figura 7.5 mostra i risultati ottenuti. Sul Launchpad è possibile notare che l'avvio del servizio di rete `nsd-enforce-untrusted` è fallito e viene rilevato dopo un tempo di circa 5 minuti. Tale costante temporale è impostata all'interno del progetto Open Source MANO e specifica il tempo limite entro cui lo stato di un servizio di rete dovrebbe diventare attivo, se il timeout scade allora c'è stato qualche problema durante la sua creazione. La Trust Dashboard mostra che sia il nodo Controller sia il nodo Compute hanno un'attestazione valida. L'OpenStack Dashboard non ha nessun elemento da mostrare proprio perché la verifica di integrità eseguita sulle macchine virtuali è fallita quindi la policy associata di tipo hash and enforce ha impedito il loro avvio.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.6 permettono di vedere le stesse informazioni mostrate nelle interfacce grafiche. Si può notare come prima dell'avvio di ogni macchina virtuale venga verificata l'attestazione dei server fisici registrati. La verifica di

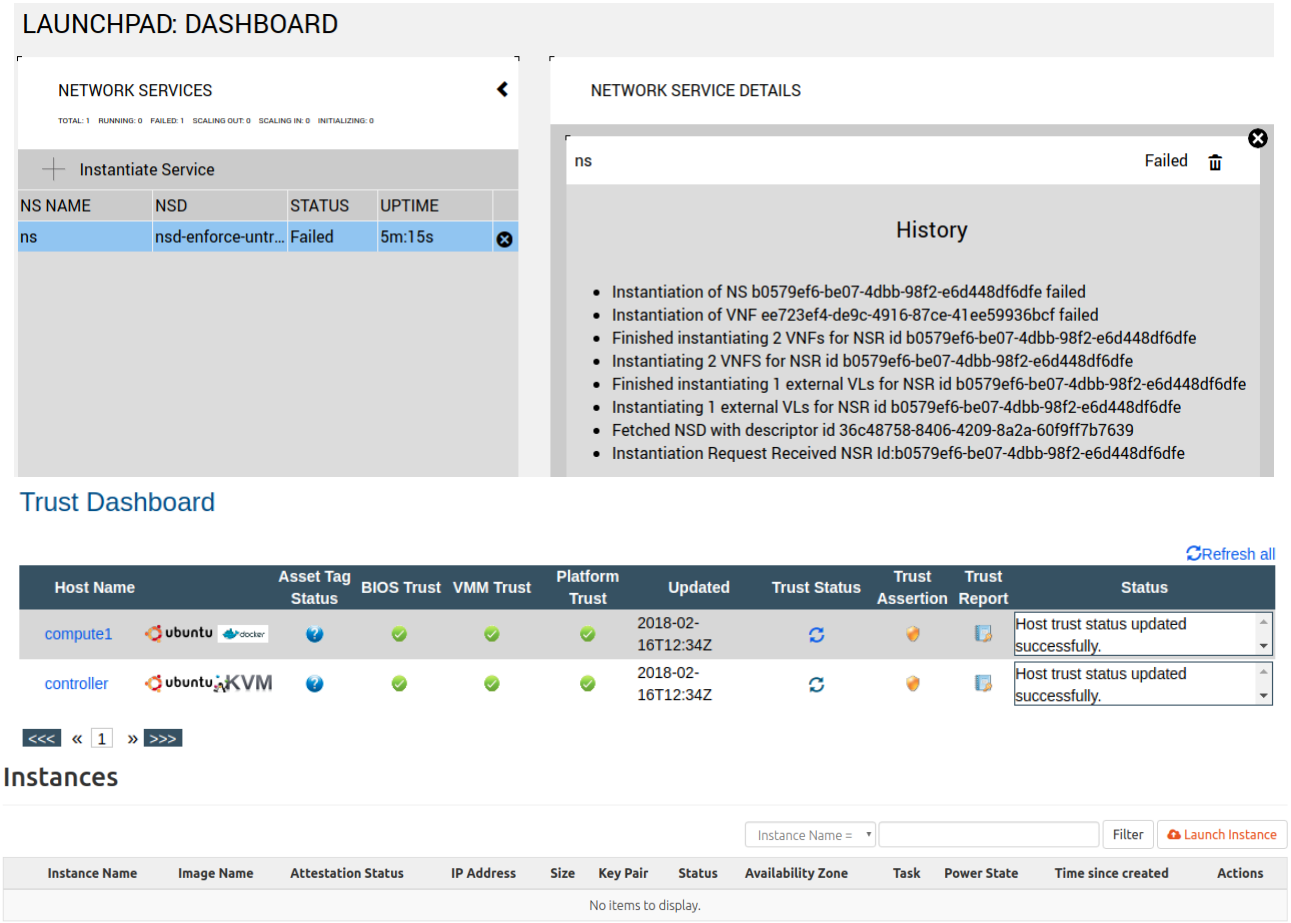

Figura 7.5. Risultati del Test VM enforce untrusted.

```
2018-02-16T12:35:29 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
    compute1(130.192.1.91) trust
2018-02-16T12:35:29 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
    controller(130.192.1.90) trust
...
2018-02-16T12:37:31 ERROR openmano.vim.openstack vimconn_openstack.py:1176
   Untrusted VM dd9394e7-1497-4dc3-8237-c2e10cc4f626 with hash and enforce
   policy!
...
2018-02-16T12:37:32 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
    compute1(130.192.1.91) trust
2018-02-16T12:37:32 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
    controller(130.192.1.90) trust
...
2018-02-16T12:39:31 ERROR openmano.vim.openstack vimconn_openstack.py:1176
   Untrusted VM 281026ec-3fed-498f-901b-02e838dd8479 with hash and enforce
   policy!
```

Figura 7.6. Informazioni relative al Test VM enforce untrusted presenti in `openmano.log`.

entrambi i server fisici va a buon fine mentre la verifica sulle macchine virtuali fallisce e viene inserito un apposito messaggio di errore all'interno del file di log.

7.6 Test VM hash trusted

Questo test simula l'avvio del servizio di rete denominato `nsd-hash-trusted`. L'immagine caricata su Glance è la `hash-encrypt-trusted` ed è associata ad una Trust policy di tipo hash only che prevede la misurazione della cartella `/boot` contenente il kernel Linux ed i principali file necessari all'avvio del sistema.

La Figura 7.7 mostra i risultati ottenuti. Sul Launchpad è possibile notare che il servizio di rete

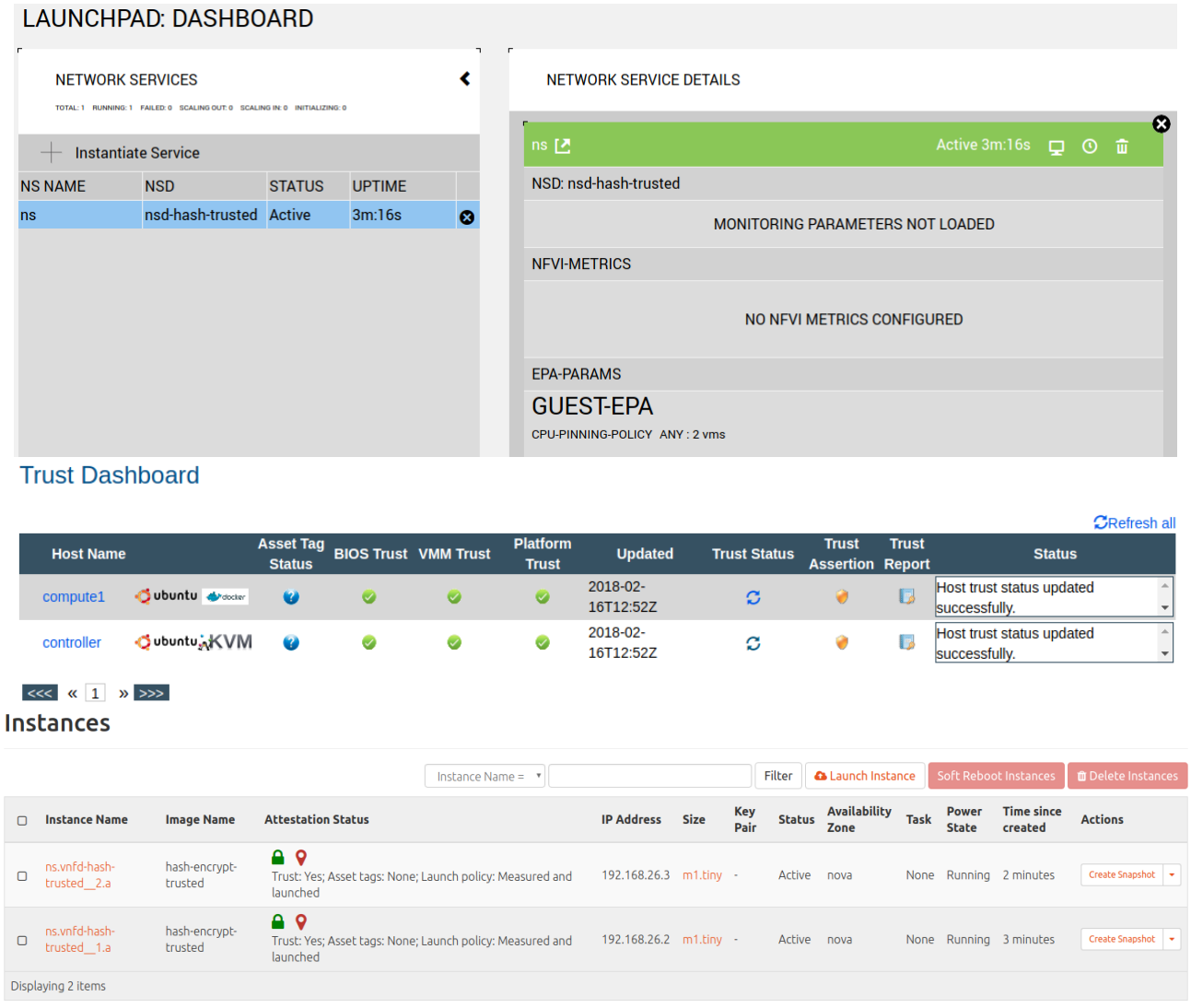

Figura 7.7. Risultati del Test VM hash trusted.

`nsd-hash-trusted` è stato avviato correttamente infatti il suo stato è attivo. La Trust Dashboard mostra che sia il nodo Controller sia il nodo Compute hanno un'attestazione valida. L'OpenStack Dashboard elenca le istanze avviate e per ognuna di esse visualizza lo stato che è fidato e la policy con cui sono state lanciate che è di tipo measured and launched inoltre assegna ad ognuna di esse un indirizzo IP privato attraverso cui possono comunicare fra loro.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.8 permettono di

```
2018-02-16T12:53:01 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     compute1(130.192.1.91) trust
2018-02-16T12:53:01 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     controller(130.192.1.90) trust
...
2018-02-16T12:54:26 DEBUG openmano.vim.openstack vimconn_openstack.py:1294 VM
    db54838b-5c87-4f73-ab83-34a5c51bb7ec trust and run with policy MeasureOnly!
...
2018-02-16T12:54:26 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     compute1(130.192.1.91) trust
2018-02-16T12:54:26 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     controller(130.192.1.90) trust
...
2018-02-16T12:55:27 DEBUG openmano.vim.openstack vimconn_openstack.py:1294 VM
    3a24be99-a38e-45a1-8551-527a351fcf02 trust and run with policy MeasureOnly!
```

Figura 7.8. Informazioni relative al Test VM hash trusted presenti in `openmano.log`.

vedere le stesse informazioni mostrate nelle interfacce grafiche. Si può notare come prima dell'avvio di ogni macchina virtuale venga verificata l'attestazione dei server fisici registrati.

7.7 Test VM hash untrusted

Questo test simula l'avvio del servizio di rete denominato `nsd-hash-untrusted`. L'immagine caricata su Glance è la `hash-encrypt-untrusted` ed è associata ad una Trust policy di tipo hash only che prevede la misurazione della cartella `/bin` contenente i programmi di sistema essenziali.

Le motivazioni che portano la cartella `/bin` ad essere considerata non fidata sono state discusse nella Sezione 7.5.

La Figura 7.9 mostra i risultati ottenuti. Sul Launchpad è possibile notare che il servizio di rete `nsd-hash-untrusted` è stato avviato correttamente infatti il suo stato è attivo. La Trust Dashboard mostra che sia il nodo Controller sia il nodo Compute hanno un'attestazione valida. L'OpenStack Dashboard elenca le istanze avviate e per ognuna di esse visualizza il fallimento delle misurazioni eseguite sulle VM quindi lo stato non è fidato ma le macchine virtuali sono comunque in esecuzione perché la policy con cui sono state avviate è di tipo hash only. Per ogni VM sarà presente anche un indirizzo IP privato attraverso cui le macchine virtuali possono comunicare fra loro.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.10 permettono di vedere le stesse informazioni mostrate nelle interfacce grafiche. Si può notare come prima dell'avvio di ogni macchina virtuale venga verificata l'attestazione dei server fisici registrati. La verifica di entrambi i server fisici va a buon fine mentre la verifica sulle macchine virtuali fallisce ma le macchine virtuali vengono avviate ugualmente e all'interno del file di log viene inserito un apposito messaggio di warning in cui viene indicata questa situazione di potenziale insicurezza.

7.8 Test Compute untrusted

Nei test precedenti sono stati valutati i casi di macchine virtuali fidate o non fidate e con policy differenti. Questo test tratta il caso in cui uno dei server fisici, il nodo Compute, si trovi in uno stato non fidato. Durante la realizzazione della Trust policy per il nodo Compute sono stati presi come riferimento i file legati all'installazione Open CIT ed è stato aggiunto un ulteriore file denominato

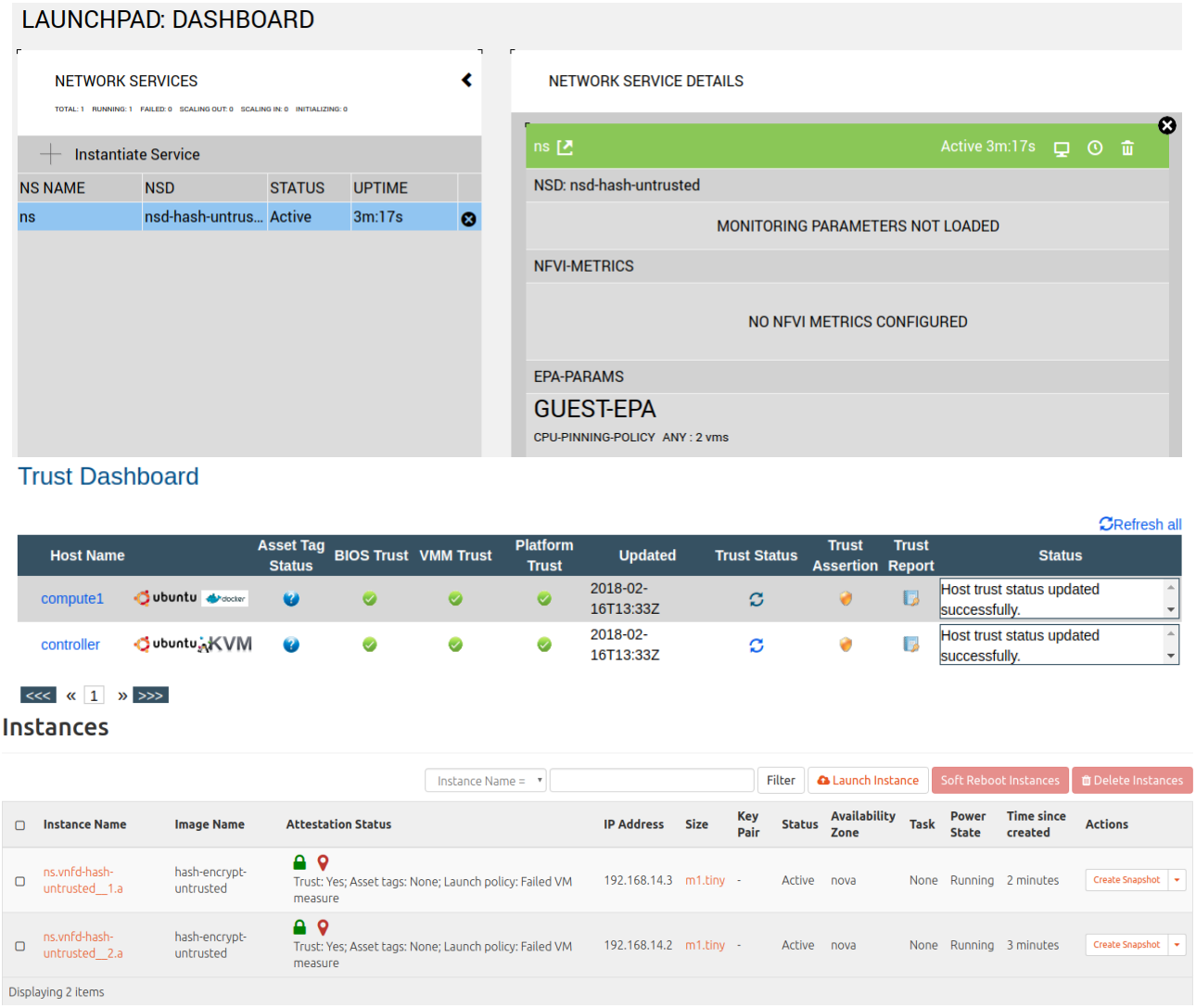

Figura 7.9. Risultati del Test VM hash untrusted.

`/trust-untrust` nella cartella di root, per la realizzazione di questo test è stato modificato il contenuto del file ed è stato riavviato il nodo Compute.

La Figura 7.11 mostra i risultati ottenuti. Il servizio di rete utilizzato è `nsd-enforce-trusted` ma sarebbe stato possibile utilizzare qualsiasi altro servizio di rete e il risultato sarebbe stato lo stesso. Sul Launchpad è possibile notare che l'avvio del servizio di rete `nsd-enforce-trusted` è fallito e viene rilevato dopo un tempo di circa 5 minuti. La Trust Dashboard mostra per il nodo Compute un problema di attestazione legato alle VMM MLE e quindi la piattaforma nel suo complesso non può essere considerata fidata. L'OpenStack Dashboard non mostra nessuna istanza, in realtà nemmeno si arriva alla valutazione dell'integrità delle macchine virtuali. Dopo il rilevamento dello stato non fidato del nodo Compute verrà lanciata un'eccezione che si occuperà di bloccare lo sviluppo del servizio di rete e di rilasciare le risorse occupate.

Nel caso di attestazioni legate ad un server fisico è possibile visualizzare quali siano i file o le cartelle coinvolte nel problema di attestazione. La Figura 7.12 mostra i registri PCR coinvolti ed evidenzia l'errore all'interno del PCR 19, in particolare nel componente `tbootxm`. Accedendo al dettaglio sarà possibile notare che il file il cui contenuto non corrisponde alla misura memorizzata come riferimento è il file `/trust-untrust`.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.13 permettono di vedere come il nodo Compute sia il primo server fisico ad essere valutato, rilevato il suo stato

```
2018-02-16T13:34:21 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     compute1(130.192.1.91) trust
2018-02-16T13:34:21 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     controller(130.192.1.90) trust
...
2018-02-16T13:36:01 WARNING openmano.vim.openstack vimconn_openstack.py:1292
    VM 27babc8e-7d40-47e3-add7-29b899031a06 untrust but run because the policy
    is MeasureOnly
...
2018-02-16T13:36:01 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     compute1(130.192.1.91) trust
2018-02-16T13:36:01 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
     controller(130.192.1.90) trust
...
2018-02-16T13:37:01 WARNING openmano.vim.openstack vimconn_openstack.py:1292
    VM 2fbe6a1a-54a4-49db-8408-efc690e9e625 untrust but run because the policy
    is MeasureOnly
```

Figura 7.10. Informazioni relative al Test VM hash untrusted presenti in `openmano.log`.

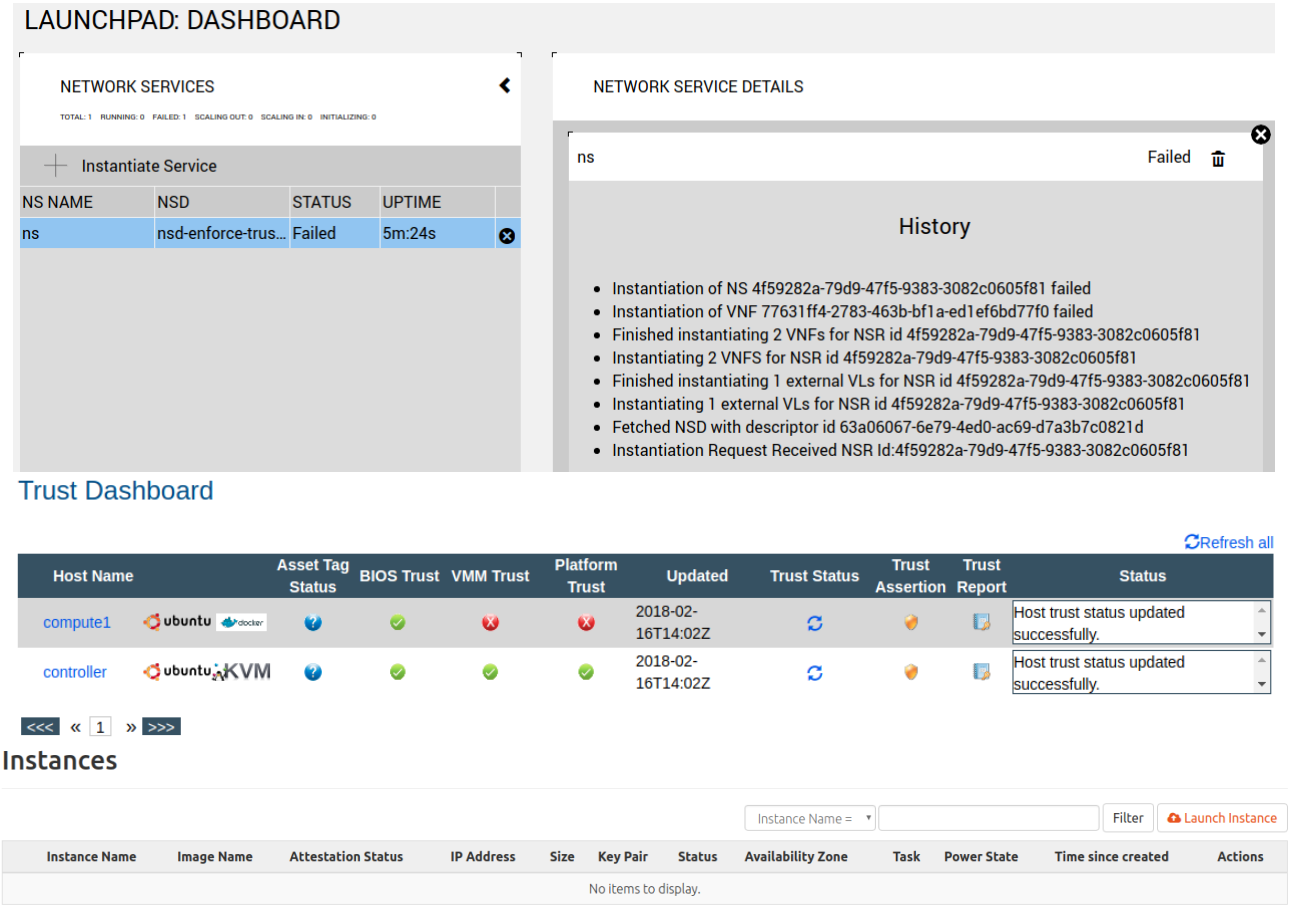

Figura 7.11. Risultati del Test Compute untrusted.

non fidato viene inserito un messaggio di errore all'interno del log dopo il quale sarà lanciata

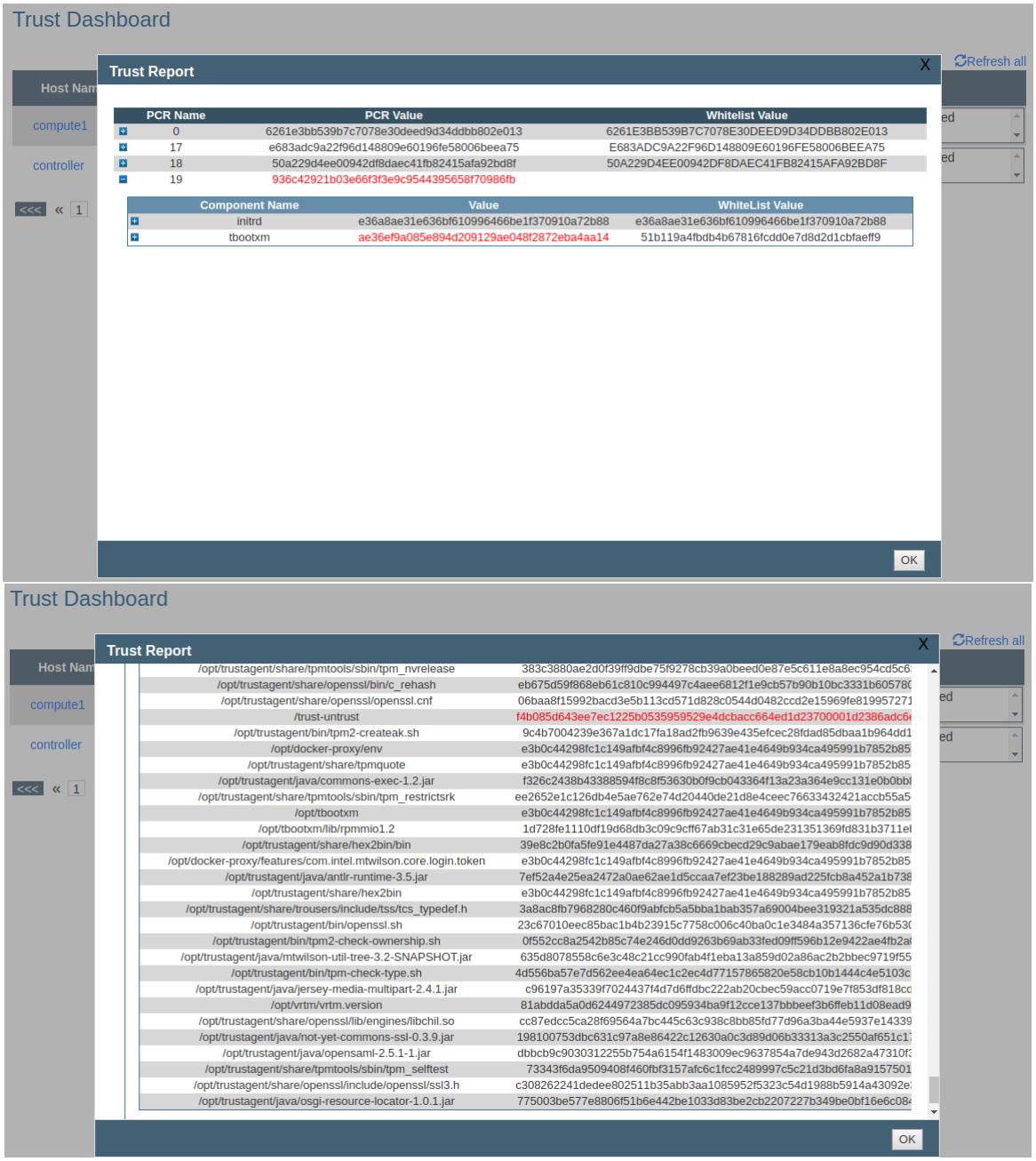

Figura 7.12. Trust Report del Test Compute untrusted.

```
2018-02-16T14:05:27 ERROR openmano.vim.openstack vimconn_openstack.py:954 Host
    'compute1(130.192.1.91)' untrust: impossible to instantiate vm
```

Figura 7.13. Informazioni relative al Test Compute untrusted presenti in `openmano.log`.

un'eccezione.

7.9 Test Controller untrusted

Questo test tratta il caso in cui uno dei server fisici, il nodo Controller, si trovi in uno stato non fidato. Durante la realizzazione della Trust policy per il nodo Controller sono stati presi come riferimento i file legati all'installazione Open CIT ed è stato aggiunto un ulteriore file denominato `/trust-untrust` nella cartella di root, per la realizzazione di questo test è stato modificato il contenuto del file ed è stato riavviato il nodo Controller.

La Figura 7.14 mostra i risultati ottenuti. Il servizio di rete utilizzato è `nsd-enforce-trusted`

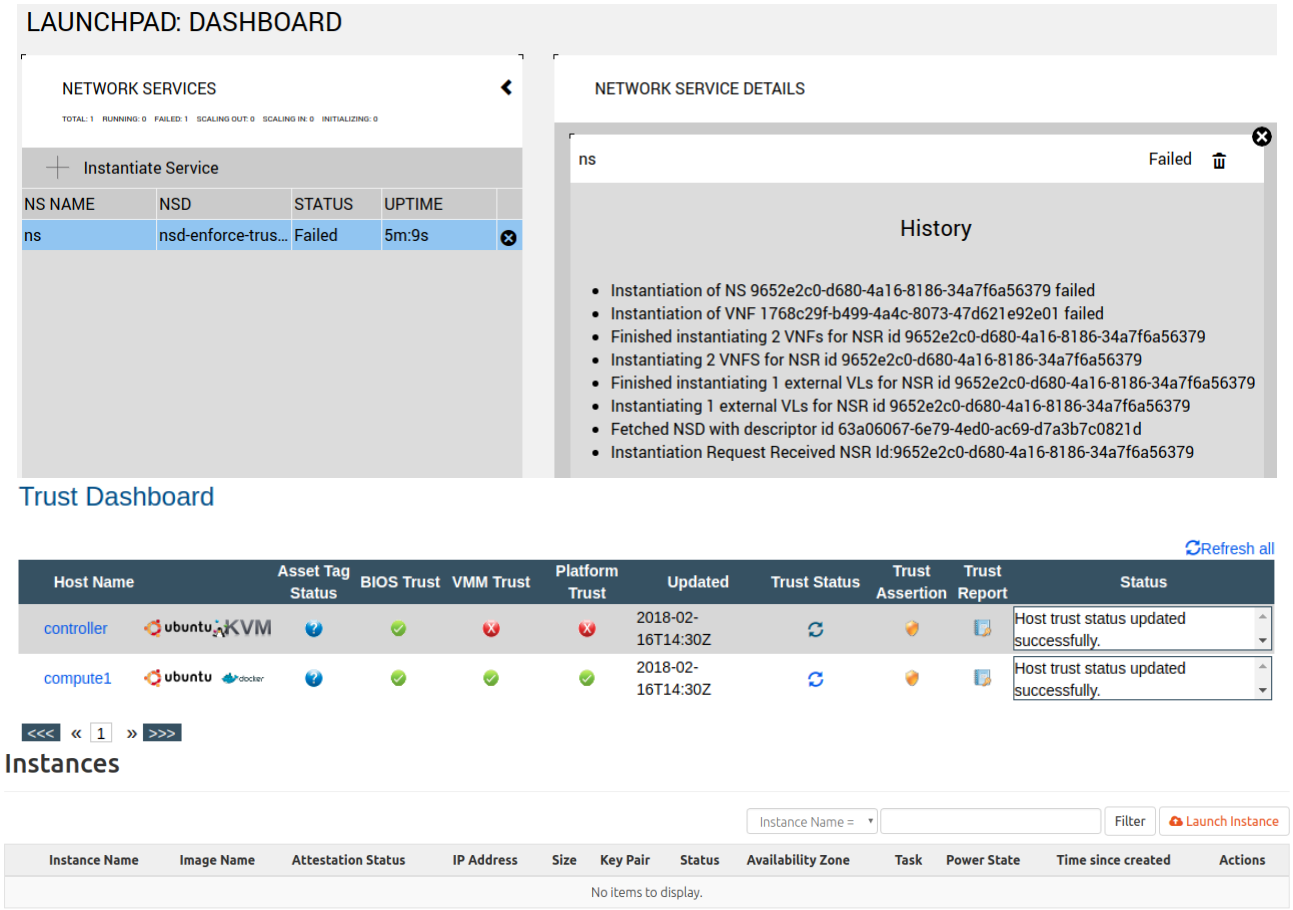

Figura 7.14. Risultati del Test Controller untrusted.

ma sarebbe stato possibile utilizzare qualsiasi altro servizio di rete e il risultato sarebbe stato lo stesso. Sul Launchpad è possibile notare che l'avvio del servizio di rete `nsd-enforce-trusted` è fallito e viene rilevato dopo un tempo di circa 5 minuti. La Trust Dashboard mostra per il nodo Controller un problema di attestazione legato alle VMM MLE e quindi la piattaforma nel suo complesso non può essere considerata fidata. L'OpenStack Dashboard non mostra nessuna istanza, in realtà nemmeno si arriva alla valutazione dell'integrità delle macchine virtuali. Dopo il rilevamento dello stato non fidato del nodo Controller verrà lanciata un'eccezione che si occuperà di bloccare lo sviluppo del servizio di rete e di rilasciare le risorse occupate.

Nel caso di attestazioni legate ad un server fisico è possibile visualizzare quali siano i file o le cartelle coinvolte nel problema di attestazione. La Figura 7.15 mostra i registri PCR coinvolti ed evidenzia l'errore all'interno del PCR 19, in particolare nel componente `tbootxm`. Accedendo al dettaglio sarà possibile notare che il file il cui contenuto non corrisponde alla misura memorizzata come riferimento è il file `/trust-untrust`.

I dati relativi a questo test estratti da `openmano.log` e mostrati nella Figura 7.16 permettono

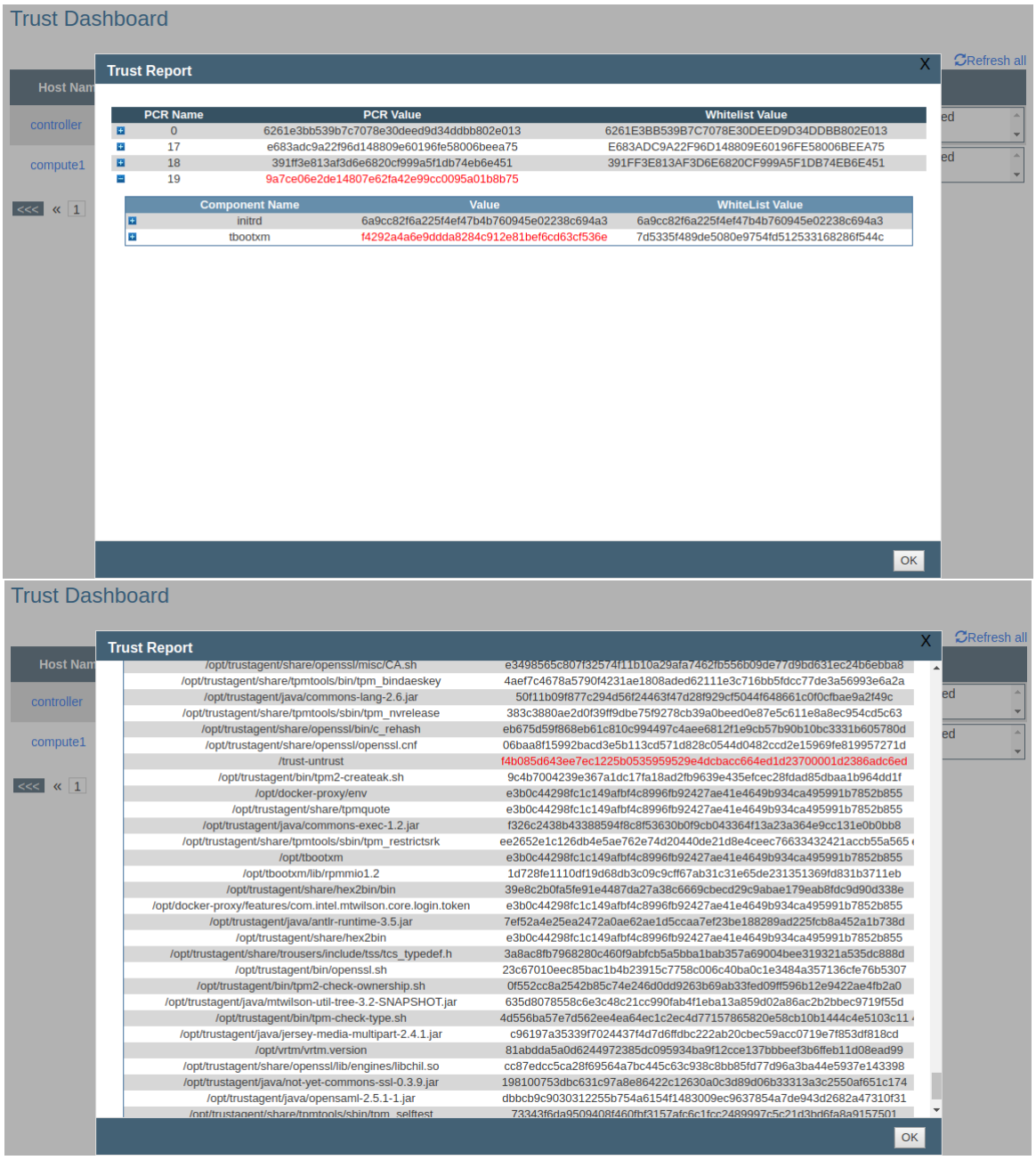

Figura 7.15. Trust Report del Test Controller untrusted.

```
2018-02-16T14:32:55 DEBUG openmano.vim.openstack vimconn_openstack.py:957 Host
    compute1(130.192.1.91) trust
2018-02-16T14:32:55 ERROR openmano.vim.openstack vimconn_openstack.py:954 Host
    'controller(130.192.1.90)' untrust: impossible to instantiate vm
```

Figura 7.16. Informazioni relative al Test Controller untrusted presenti in `openmano.log`.

di vedere come il nodo Compute sia il primo server fisico ad essere valutato e viene attestato correttamente mentre lo stato del nodo Controller non è considerato fidato quindi viene inserito

un messaggio di errore all'interno del log dopo il quale sarà lanciata un'eccezione.

7.10 Commenti ai test

Nelle sezioni precedenti sono stati illustrati dei test relativi all'avvio di un servizio di rete in scenari differenti. Il sistema coinvolto, con le modifiche apportate ad Open Source MANO, è stato in grado di reagire correttamente ai problemi che si sono verificati tenendo traccia dei risultati ottenuti all'interno di un file di log. OSM ha rilevato sia i casi in cui un nodo fisico non è fidato ed ha impedito l'avvio di qualsiasi servizio di rete sia i casi in cui l'immagine delle macchine virtuali da avviare è non integra ed in quest'ultimo caso ha adottato un comportamento differente a seconda della policy associata all'immagine. È stato impedito l'avvio della macchina virtuale se la policy era hash and enforce mentre è stato consentito ugualmente il suo avvio se la policy era hash only.

La Tabella 7.1 riassume le principali caratteristiche con cui è stato avviato ogni test ed i relativi risultati ottenuti.

La soluzione proposta consente di verificare direttamente in Open Source MANO le caratteristiche legate all'attestazione e basandosi su di esse è in grado di prendere le opportune decisioni. L'integrazione fra OSM ed Open CIT non può essere considerata totale infatti non sono direttamente disponibili in OSM le funzionalità necessarie per creare le Trust policy virtualizzate e non virtualizzate e per registrare gli host fisici su cui dovrà essere eseguita l'attestazione.

Nome test	*Controller*	*Compute*	*File controllati Host*	*Policy virtualizzata*	*File controllati VM*	*Risultato VM*	*Risultato NS*

VM enforce trusted	fidato	fidato	`/opt/mtwilson/*`
			`/opt/tbootxm/*`
			`/opt/trustagent/*`
			`/opt/openstack-ext/*`
			`/opt/policyagent/*`
			`/opt/vrtm/*`
			`/opt/docker-proxy/*`
			`/trust-untrust`
				hash and enforce	`/boot/*`	Avviate con stato fidato	Avviato correttamente

VM enforce untrusted	fidato	fidato		hash and enforce	`/bin/*`	Errore - stato non fidato	Non avviato
VM hash trusted	fidato	fidato		hash only	`/boot/*`	Avviate con stato fidato	Avviato correttamente
VM hash untrusted	fidato	fidato		hash only	`/bin/*`	Avviate con stato non fidato	Avviato correttamente
Compute untrusted	fidato	non fidato		X	X	X	Non avviato
Controller untrusted	non fidato	fidato		X	X	X	Non avviato

Tabella 7.1. Tabella riepilogativa dei Test eseguiti.

Capitolo 8

Conclusioni

Questo lavoro di tesi si è occupato dell'estensione delle funzionalità offerte da Open Source MANO per supportare anche l'attestazione remota. Il meccanismo classico è stato arricchito e reso più sicuro grazie alla verifica preventiva dello stato di fiducia in cui si trovano i server fisici presenti nell'infrastruttura ed offrendo anche la possibilità di controllare lo stato di integrità delle macchine virtuali prima di decidere di avviare il servizio di rete. Queste caratteristiche hanno permesso di rafforzare la sicurezza offerta e gestita dall'orchestratore del servizio di rete.

Un lavoro preliminare alla realizzazione della soluzione è stato quello di analizzare nel dettaglio il funzionamento ed i requisiti hardware necessari per offrire le funzionalità di attestazione remota e per permettere la creazione di un'infrastruttura virtuale che potesse essere pilotata dall'orchestratore di rete. Un altro passo verso la realizzazione di una soluzione affidabile è stato l'identificazione dei software più adatti per offrire un servizio del tipo IaaS attraverso cui è stato possibile astrarsi dai dettagli fisici dell'hardware presente, per offrire le caratteristiche di attestazione remota e per permettere ad un unico orchestratore di rete di gestire in maniera ottimale e sicura l'avvio di nuovi servizi di rete. L'analisi di ognuno dei software utilizzati ha fatto emergere dei vincoli che è stato necessario tenere in considerazione nella realizzazione della soluzione finale, ad esempio Open CIT ha richiesto la presenza di macchine con TPM ed Intel TXT abilitato e la necessità di utilizzare OpenStack come IaaS.

La scelta di utilizzare tre macchine fisiche distinte è stata dettata dalla necessità di installare direttamente sul server fisico che dovrà essere attestato il Trust Agent Open CIT per permettere l'estensione della catena di fiducia fin dalla fase di avvio. L'utilizzo delle tre macchine ha voluto rappresentare una simulazione più vicina, seppur limitata, alle architetture reali che potrebbero essere sviluppate. Le architetture reali spesso posseggono server distinti su cui eseguire le computazioni ed un controllore che permette di pilotare il loro funzionamento. Per motivi logistici si è scelta un'unica macchina come nodo per la computazione ma la soluzione sarebbe facilmente estendibile registrando all'interno del codice l'hostname e l'indirizzo IP di altri compute node. Negli scenari reali tutti gli altri componenti di Open CIT ed OSM sarebbero distribuiti su più server e si troverebbero in macchine diverse da quelle utilizzate per implementare il servizio cloud. La scelta è stata proprio quella di mantenere la separazione dal controller e dal compute node realizzando le loro funzionalità su una terza macchina fisica dotata di maggiori risorse computazionali per offrire un buon livello di prestazioni ad ogni servizio che su di essa è implementato.

La realizzazione di un'architettura su cui andare a svolgere dei test ha previsto una fase di studio dell'architettura ETSI NFV, dei software OpenStack ed Open Source MANO, dei componenti necessari per implementare l'attestazione all'interno dell'architettura cioè dei requisiti di installazione di Open CIT. È stato necessario approfondire anche l'integrazione e la comunicazione dei tre software in modo da metterli nelle condizioni di interagire nella maniera migliore possibile.

I test sono stati realizzati tenendo presente i possibili scenari che si possono verificare durante l'avvio di un servizio di rete e gestendo i casi particolari e le eccezioni in modo da lasciare l'architettura in uno stato consistente per i successivi utilizzi. Sono state create nuove eccezioni per indicare il funzionamento anomalo in alcuni casi e sono stati aggiunti dei messaggi che permettono

di tracciare esattamente il comportamento adottato dall'orchestratore di rete. La soluzione tiene conto dei casi in cui un server fisico non ha un'attestazione valida ma anche di quelli in cui è l'integrità delle immagini legate alle macchine virtuali da avviare a fallire e le gestisce in base alle politiche associate ad esse. I risultati hanno dimostrato la possibilità di verificare lo stato di un servizio di rete in maniera semplice tuttavia nel caso di attestazioni non valide e quindi quando è necessario impedire l'avvio del servizio di rete il tempo necessario a rilevarlo sembra eccessivo, un'integrazione diretta fra Open Source MANO ed i componenti Open CIT potrebbe migliorare il tempo di risposta del sistema in queste situazioni.

La soluzione realizzata si propone come uno strumento facilmente utilizzabile dal gestore dei servizi di rete che può ricevere le informazioni sul livello di fiducia dei server fisici e delle macchine virtuali che costituiscono il servizio di rete semplicemente consultando il file di log presente nel Resource Orchestrator.

Il lavoro presentato in questa tesi può essere oggetto di futuri miglioramenti. Dal punto di vista prestazionale si potrebbero realizzare architetture maggiormente distribuite con server dedicati ad ogni singolo servizio e ciò porterebbe a dei tempi di risposta molto più bassi. Un altro punto da migliorare è l'integrazione in Open Source MANO delle funzionalità attualmente gestibili solo tramite i componenti Open CIT come la creazione di policy, il caricamento delle immagini su Glance e la registrazione di nuovi server per l'attestazione. Si potrebbe pensare a semplificare ulteriormente la visualizzazione delle informazioni immagazzinate nel log al fine di rendere più trasparente lo stato attuale a chi gestisce il servizio di rete, ciò sarebbe possibile attraverso modifiche da apportare all'interfaccia grafica su cui potrebbero essere riportate le informazioni associate all'attestazione di server fisici e macchine virtuali.

Anche i software utilizzati hanno delle limitazioni, una su tutti è la necessità di attestazione a tempo di esecuzione cioè nel caso di server fisici l'estensione della catena di fiducia ed il calcolo delle misurazioni avviene solamente all'avvio del server, se lo stato del server è fidato non verrà rilevato nessun tentativo di manomissione o modifiche ai file ed alle cartelle del server fino al suo successivo riavvio. Se l'Attestation Server richiedesse una nuova attestazione il server manderebbe sempre le stesse misure che aveva acquisito all'avvio invece sarebbe molto più utile e sicuro poter usufruire di un meccanismo che riesca ad attestare periodicamente il server in modo da rilevare in breve tempo i tentativi di manomissioni al fine di attuare le contromisure necessarie.

Un'ulteriore area su cui indagare è la possibilità, in Open Source MANO, di automatizzare l'acquisizione degli hostname e degli indirizzi IP dei compute node presenti in un certo data center. La conoscenza di questi dati è fondamentale per recuperare la Trust assertion dall'Attestation Server con cui è possibile stabilire se il server è fidato. La soluzione proposta in questo lavoro di tesi prevede la registrazione manuale dei server che dovranno essere verificati ma un meccanismo automatico semplificherebbe il lavoro del gestore dei servizi di rete e renderebbe il meccanismo implementato più flessibile ed adattabile a nuove configurazioni e nuovi scenari.

Bibliografia

[1] RightScale Cloud Management, https://www.rightscale.com/

[2] RightScale, "State of the Cloud Report", 2018, https://assets.rightscale.com/uploads/pdfs/RightScale-2018-State-of-the-Cloud-Report.pdf

[3] Trusted computing, https://it.wikipedia.org/wiki/Trusted_computing

[4] University of Cambridge, "Trusted computing Frequently Asked Questions, Version 1.1", August 2003, http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html

[5] Trusted Computing Group, "TPM 2.0: A Brief Introduction", October 2016, https://www.trustedcomputinggroup.org/wp-content/uploads/TPM-2.0-A-Brief-Introduction.pdf

[6] Trusted Computing Group, "TCG Architecture Overview, Version 1.4", August 2007, https://trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf

[7] Trusted Computing Group, "TCG Software Stack (TSS) Specification Version 1.2 Level 1", January 2006, https://trustedcomputinggroup.org/wp-content/uploads/TSS_Version_1.2_Level_1_FINAL.pdf

[8] G.Coker, J.Guttman, P.Loscocco, A.Herzog, J.Millen, B.O'Hanlon, J.Ramsdell, A.Segall, J.Sheehy, B.Sniffen, "Principles of Remote Attestation", International Journal of Information Security, Vol. 10, June 2011, pp. 63-81, DOI 10.1007/s10207-011-0124-7

[9] Trusted Computing Group, "TCPA Main Specification Version 1.1b", February 2002, https://trustedcomputinggroup.org/wp-content/uploads/TCPA_Main_TCG_Architecture_v1_1b.pdf

[10] C.J.Bare, "Attestation and Trusted Computing", CSEP 590: Practical Aspects of Modern Cryptography, March 2006, https://courses.cs.washington.edu/courses/csep590/06wi/finalprojects/bare.pdf

[11] J.Camenisch, "Direct Anonymous Attestation Explained", IBM Research, https://pdfs.semanticscholar.org/4cc6/257c4b2d649a09b08c1789dcaf3aeb3c1fda.pdf

[12] Trusted Computing Group, "Trusted Platform Module Library Part 1: Architecture", October 2014, https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.16.pdf

[13] W.Arthur, D.Challener, K.Goldman "Hierarchies" nel libro "A Practical Guide to TPM 2.0" Apress, 2015, pp. 105-118, DOI 10.1007/978-1-4302-6584-9

[14] L.Chen, J.Li, "Flexible and scalable digital signatures in TPM 2.0", CCS '13 Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, Berlin (Germany), November 04-08, 2013, pp. 37-48, DOI 10.1145/2508859.2516729

[15] "Tecnologie di virtualizzazione", https://it.wikiversity.org/wiki/Tecnologie_di_virtualizzazione

[16] VMware ESXi, https://www.vmware.com/products/esxi-and-esx.html

[17] Microsoft, https://www.microsoft.com

[18] The Xen Project, https://www.xenproject.org/

[19] KVM, https://www.linux-kvm.org/page/Main_Page

[20] VMware Workstation Player, https://www.vmware.com/it/products/workstation-player.html

[21] VirtualBox, https://www.virtualbox.org/

[22] S.Conroy, "History of Virtualization", https://www.idkrtm.com/history-of-virtualization/

[23] A.Capodaglio, "Introduzione alle virtualizzazioni", http://www.biblioteche.unibo.it/capodaglio/informatica/materiale/virtualizzazioni1.pdf
[24] ETSI, "Network Functions Virtualisation-Introductory White Paper", October 2012, https://portal.etsi.org/NFV/NFV_White_Paper.pdf
[25] F.Risso, "Introduction to software defined networks", https://c7f227f8-a-340fb901-s-sites.googlegroups.com/a/frisso.net/par/files/SDNintro.pdf
[26] F.Risso, "OpenFlow", https://c7f227f8-a-340fb901-s-sites.googlegroups.com/a/frisso.net/par/files/OpenFlow.pdf
[27] ETSI, "Network Functions Virtualisation (NFV); Architectural Framework", December 2014, https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV002v1.2.1-GS-NFVArchitecturalFramework.pdf
[28] ETSI, "Network Functions Virtualisation (NFV); Virtualisation Requirements", October 2013, https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV004v1.1.1-GS-NFVVirtualisationRequirements.pdf
[29] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", December 2014, https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-MAN001v1.1.1-GS-ManagementandOrchestration.pdf
[30] S.Lal, T.Taleb, A.Dutta, "NFV: Security Threats and Best Practices", IEEE Communications Magazine, Vol. 55, May 2017, pp. 211-217, DOI 10.1109/MCOM.2017.1600899
[31] ETSI, "Network Functions Virtualisation (NFV); Trust; Report on Attestation Technologies and Practices for Secure Deployments", October 2017, https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SEC007v1.1.1-GR-NFVAttestationreport.pdf
[32] L.Grossi, E.Maffione, G.Marasso, S.Ruffino, "SDN e NFV: quali sinergie?", Notiziario Tecnico Telecom Italia, No. 2, 2014, http://www.telecomitalia.com/content/dam/telecomitalia/it/archivio/documenti/Innovazione/MnisitoNotiziario/2014/2-2014/capitolo-5/SDNeNFVqualisinergie.pdf
[33] OpenStack, https://www.openstack.org/
[34] OpenStack, le ragioni di un successo, https://www.digital4trade.it/tech-lab/openstack-le-ragioni-di-un-successo/
[35] Open Source MANO, https://osm.etsi.org
[36] Cloud-init, http://cloudinit.readthedocs.io/
[37] YAML, http://yaml.org/
[38] OpenVIM, https://github.com/nfvlabs/openvim
[39] VMware vCloud Director, https://www.vmware.com/products/vcloud-director.html
[40] Amazon EC2, https://aws.amazon.com/it/ec2/
[41] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC-6020, October 2010, DOI 10.17487/RFC6020
[42] Open Cloud Integrity Technology, https://01.org/opencit
[43] T.Knoll, Intel, "Intel Trusted Execution Technology and trust attestation overview", https://ci.spdk.io/download/events/2017-summit/09_-_Day_2_-_Knoll_and_Natesan_-_Builders_Dev_Summit_OTA-CIT_Presentation.pdf
[44] S.Orrin, S.Forage, Red Hat Summit, "Delivering Trusted Clouds", https://www.redhat.com/files/summit/session-assets/2016/SS88843-delivering-trusted-clouds_how-intel-and-red-hat-integrated-solutions-for-secure-cloud-computing.pdf
[45] Open CIT 3.2.1 Product Guide, https://github.com/opencit/opencit/wiki/Open-CIT-3.2.1-Product-Guide
[46] G.Lavado, OpenStack Summit, "(A true story on) Achieving end-to-end NFV with OpenStack and Open Source MANO", https://www.openstack.org/assets/presentation-media/Achieving-end-to-end-NFV-with-OpenStack-and-Open-Source-MANO.pdf
[47] D.Andrushko, G.Elkinbard, Mirantis, "What is the best NFV Orchestration platform? A review of OSM, Open-O, CORD, and Cloudify", March 2017, https://www.mirantis.com/blog/which-nfv-orchestration-platform-best-review-osm-open-o-cord-cloudify/
[48] LXD, https://linuxcontainers.org/lxd/
[49] LXC, https://linuxcontainers.org/lxc/
[50] Open Baton, https://openbaton.github.io/

[51] Open Network Automation Platform, https://www.onap.org/
[52] Cloudify, https://cloudify.co/
[53] OpenMANO, http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab/openmano
[54] A.Tierno, G.Garcia, F.J.Ramon, P.Montes, A.Lopez, Telefonica, "MWC Demo Components Telefonica OpenMANO", https://osm.etsi.org/wikipub/images/2/OSM(16)000020_MWC_demo_components_-_OpenMANO.pdf
[55] BusyBox, https://busybox.net/

Appendice A

Manuale utente

In questo manuale presenteremo una guida per l'installazione di OpenStack nella Sezione A.1, per l'installazione di Open Source MANO nella Sezione A.2, per l'installazione di Open CIT nella Sezione A.3, per la configurazione del firewall nella Sezione A.4. Infine verrà spiegato come è possibile applicare la soluzione proposta in questa tesi nella Sezione A.5.

A.1 Installazione e configurazione OpenStack

Questa guida è estratta dalla documentazione ufficiale OpenStack [33]. La versione utilizzata per l'installazione è OpenStack Mitaka, l'architettura di riferimento è costituita da due macchine fisiche distinte ognuna delle quali è caratterizzata da CPU Intel i7 di terza generazione, 8 GB di RAM e sistema operativo Ubuntu 16.04.2 LTS. Una macchina svolgerà il compito di controller node e l'altra sarà l'unico compute node con cui è stata realizzata l'architettura.

In questo manuale, nella descrizione delle configurazioni dei vari componenti, si indicherà nella forma `SERVIZIO_PASS` la necessità di sostituire quel valore con la password generata per quel servizio e con `SERVIZIO_DBPASS` la necessità di sostituire quel valore con la password generata per il database di quel servizio.

A.1.1 Configurazione della rete

Sia per il controller node sia per il compute node bisogna impostare un'interfaccia di gestione attraverso cui sarà possibile comunicare con i vari componenti OpenStack ed un'altra interfaccia si dovrà collegare alla rete del fornitore di servizi, inoltre si dovrà configurare la risoluzione dei nomi logici con i corretti indirizzi IP.

Controller node

L'host possiede una sola interfaccia di rete col nome `enp0s25`, quindi come seconda interfaccia se ne utilizzerà una virtuale. Si accede al file `/etc/network/interfaces` e si inserisce:

```
auto enp0s25
iface enp0s25 inet static
address 130.192.1.90
netmask 255.255.255.192
gateway 130.192.1.126
dns-nameservers 130.192.3.21 8.8.8.8

auto enp0s25:1
iface enp0s25:1 inet manual
```

```
up ip link set dev $IFACE up
down ip link set dev $IFACE down
```

Si accede al file `/etc/hosts` e si inserisce:

```
# controller
130.192.1.90 controller

# compute1
130.192.1.91 compute1
```

Adesso si può riavviare il sistema per rendere attive le modifiche.

Compute node

L'host possiede una sola interfaccia di rete col nome `enp0s25`, quindi come seconda interfaccia se ne utilizzerà una virtuale. Si accede al file `/etc/network/interfaces` e si inserisce:

```
auto enp0s25
iface enp0s25 inet static
address 130.192.1.91
netmask 255.255.255.192
gateway 130.192.1.126
dns-nameservers 130.192.3.21 8.8.8.8

auto enp0s25:1
iface enp0s25:1 inet manual
up ip link set dev $IFACE up
down ip link set dev $IFACE down
```

Si accede al file `/etc/hosts` e si inserisce:

```
# controller
130.192.1.90 controller

# compute1
130.192.1.91 compute1
```

Adesso si può riavviare il sistema per rendere attive le modifiche.

A.1.2 Configurazione NTP (Network Time Protocol)

È necessario sincronizzare i servizi fra i vari nodi.

Controller node

Si installa il package chrony:

```
# apt-get install chrony
```

Si modifica il file `/etc/chrony/chrony.conf` aggiungendo:

```
server 0.it.pool.ntp.org iburst
server 1.it.pool.ntp.org iburst
allow compute1
```

Adesso si può riavviare il servizio:

```
# service chrony restart
```

Compute node

Si installa il package chrony:

```
# apt-get install chrony
```

Si modifica il file `/etc/chrony/chrony.conf` aggiungendo:

```
server controller iburst
```

Adesso si può riavviare il servizio:

```
# service chrony restart
```

A.1.3 Configurazione package OpenStack

Questa configurazione deve essere eseguita sia sul controller node sia sul compute node:

```
# apt-get install software-properties-common
# add-apt-repository cloud-archive:mitaka
# apt-get update && apt-get dist-upgrade
# apt-get install python-openstackclient
```

A.1.4 Configurazione ambiente

Adesso è necessario configurare il database, la coda dei messaggi e la cache per i token. Questa configurazione deve essere eseguita sul controller node:

```
# apt-get install mariadb-server python-pymysql
```

Si crea e modifica il file `/etc/mysql/mariadb.conf.d/50-server.cnf`:

```
[mysqld]
bind-address = 130.192.1.90
default-storage-engine = innodb
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

In seguito si riavvia il servizio e si esegue lo script di installazione con i seguenti comandi:

```
# service mysql restart
# mysql_secure_installation
```

Si installa la coda di messaggi:

```
# apt-get install rabbitmq-server
# rabbitmqctl add_user openstack RABBIT_PASS
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Si installa la cache per i token:

```
# apt-get install memcached python-memcache
```

Si modifica il file `/etc/memcached.conf` con la seguente linea:

```
-l 130.192.1.90
```

Infine si riavvia il servizio:

```
# service memcached restart
```

A.1.5 Identity service (Keystone)

Questo servizio deve essere installato nel controller node.

Creazione database

Si accede come utente root e si crea il database:

```
$ mysql -u root -p

mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
  IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Si genera un valore casuale da utilizzare come token temporaneo con:

```
$ openssl rand -hex 10
```

Si disabilita la partenza automatica di Keystone e si installano i package con:

```
# echo "manual" > /etc/init/keystone.override
# apt-get install keystone apache2 libapache2-mod-wsgi
```

Si modifica il file `/etc/keystone/keystone.conf`:

```
[DEFAULT]
admin_token = ADMIN_TOKEN

[database]
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone

[token]
provider = fernet
```

dove `ADMIN_TOKEN` sarà sostituito dal valore generato precedentemente. Si popola il database:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
# keystone-manage fernet_setup --keystone-user keystone \
  --keystone-group keystone
```

Configurazione server Apache HTTP

Si modifica il file `/etc/apache2/apache2.conf` con la seguente linea:

```
ServerName controller
```

Si crea il file `/etc/apache2/sites-available/wsgi-keystone.conf` con il seguente contenuto:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone \
      group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
```

```
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone \
      group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    ErrorLogFormat "%{cu}t %M"
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>
```

Si abilita il suo utilizzo, si finalizza l'installazione e si riavvia Keystone con:

```
# ln -s /etc/apache2/sites-available/wsgi-keystone.conf \
  /etc/apache2/sites-enabled
# service apache2 restart
# rm -f /var/lib/keystone/keystone.db
# service keystone restart
```

Creazione endpoint

Si impostano i dati per configurare il servizio:

```
$ export OS_TOKEN=ADMIN_TOKEN
$ export OS_URL=http://controller:35357/v3
$ export OS_IDENTITY_API_VERSION=3
```

Si creano gli endpoint per accedere al servizio:

```
$ openstack service create \
  --name keystone --description "OpenStack Identity" identity
$ openstack endpoint create --region RegionOne \
  identity public http://controller:5000/v3
$ openstack endpoint create --region RegionOne \
  identity internal http://controller:5000/v3
$ openstack endpoint create --region RegionOne \
  identity admin http://controller:35357/v3
```

Creazione dominio, progetti, utenti, ruoli

Si creano gli elementi principali per il progetto amministrativo:

```
$ openstack domain create --description "Default Domain" default
$ openstack project create --domain default \
  --description "Admin Project" admin
$ openstack user create --domain default \
  --password-prompt admin
$ openstack role create admin
$ openstack role add --project admin --user admin admin
```

Si creano gli elementi principali per il progetto non amministrativo:

```
$ openstack project create --domain default \
  --description "Service Project" service
$ openstack project create --domain default \
  --description "Demo Project" demo
$ openstack user create --domain default \
  --password-prompt demo
$ openstack role create user
$ openstack role add --project demo --user demo user
```

Creazione script

Si crea un file denominato `admin-openrc` con il seguente contenuto:

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Rimpiazzando `ADMIN_PASS` con la password scelta durante la creazione dell'utente amministratore. Si crea un file denominato `demo-openrc` con il seguente contenuto:

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Rimpiazzando `DEMO_PASS` con la password scelta durante la creazione dell'utente demo. Lo script potrà essere utilizzato per impostare le variabili d'ambiente prima di avviare qualsiasi altra richiesta da linea di comando, la sua attivazione si ottiene col comando:

```
$ . admin-openrc
```

nel caso del progetto demo l'attivazione si ottiene col comando:

```
$ . demo-openrc
```

A.1.6 Image service (Glance)

Questo servizio deve essere installato nel controller node.

Creazione database

Si accede come utente root e si crea il database:

```
$ mysql -u root -p

mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
  IDENTIFIED BY 'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
  IDENTIFIED BY 'GLANCE_DBPASS';
```

Creazione endpoint

Si crea il servizio e gli endpoint con:

```
$ . admin-openrc
$ openstack user create --domain default --password-prompt glance
$ openstack role add --project service --user glance admin
$ openstack service create --name glance \
  --description "OpenStack Image" image
$ openstack endpoint create --region RegionOne \
  image public http://controller:9292
$ openstack endpoint create --region RegionOne \
  image internal http://controller:9292
$ openstack endpoint create --region RegionOne \
  image admin http://controller:9292
```

Installazione e configurazione package

Si installa il package con:

```
# apt-get install glance
```

Si modifica il file `/etc/glance/glance-api.conf` nel seguente modo:

```
[database]
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor = keystone

[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]`. Si modifica il file `/etc/glance/glance-registry.conf` nel seguente modo:

```
[database]
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor = keystone
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]`. Si popola il database e si riavviano i servizi con:

```
# su -s /bin/sh -c "glance-manage db_sync" glance
# service glance-registry restart
# service glance-api restart
```

A.1.7 Compute service (Nova)

Questo servizio deve essere configurato sia sul controller node sia sul compute node.

Controller node

Si accede come utente root e si crea il database:

```
$ mysql -u root -p

mysql> CREATE DATABASE nova_api;
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
  IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
  IDENTIFIED BY 'NOVA_DBPASS';
```

Si crea il servizio e gli endpoint con:

```
$ . admin-openrc
$ openstack user create --domain default \
  --password-prompt nova
$ openstack role add --project service --user nova admin
$ openstack service create --name nova \
  --description "OpenStack Compute" compute
$ openstack endpoint create --region RegionOne \
```

```
  compute public http://controller:8774/v2.1/%\(tenant_id\)s
$ openstack endpoint create --region RegionOne \
  compute internal http://controller:8774/v2.1/%\(tenant_id\)s
$ openstack endpoint create --region RegionOne \
  compute admin http://controller:8774/v2.1/%\(tenant_id\)s
```

Si installano i package con:

```
# apt-get install nova-api nova-conductor nova-consoleauth \
  nova-novncproxy nova-scheduler
```

Si modifica il file `/etc/nova/nova.conf` nel seguente modo:

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 130.192.1.90
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[api_database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
api_servers = http://controller:9292

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]` e l'opzione `logdir` presente nella sezione `[DEFAULT]`. Si popolano i database e si riavviano i servizi con:

```
# su -s /bin/sh -c "nova-manage api_db sync" nova
# su -s /bin/sh -c "nova-manage db sync" nova
# service nova-api restart
```

```
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

Compute node

Si installa il package con:

```
# apt-get install nova-compute
```

Si modifica il file `/etc/nova/nova.conf` nel seguente modo:

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 130.192.1.91
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = NOVA_PASS

[vnc]
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html

[glance]
api_servers = http://controller:9292

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]` e l'opzione `logdir` presente nella sezione `[DEFAULT]`. Si controlla se il compute node supporta l'accelerazione hardware con:

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Se il valore restituito è zero si dovrà modificare il file `/etc/nova/nova-compute.conf` nel seguente modo:

```
[libvirt]
virt_type = qemu
```

Infine si potrà riavviare il servizio:

```
# service nova-compute restart
```

A.1.8 Networking service (Neutron)

Questo servizio deve essere configurato sia sul controller node sia sul compute node.

Controller node

Si accede come utente root e si crea il database:

```
$ mysql -u root -p

mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
  IDENTIFIED BY 'NEUTRON_DBPASS';
```

Si crea il servizio e gli endpoint con:

```
$ . admin-openrc
$ openstack user create --domain default --password-prompt neutron
$ openstack role add --project service --user neutron admin
$ openstack service create --name neutron \
  --description "OpenStack Networking" network
$ openstack endpoint create --region RegionOne \
  network public http://controller:9696
$ openstack endpoint create --region RegionOne \
  network internal http://controller:9696
$ openstack endpoint create --region RegionOne \
  network admin http://controller:9696
```

Si configura la modalità di rete self service quindi si installano i package necessari con:

```
# apt-get install neutron-server neutron-plugin-ml2 \
  neutron-linuxbridge-agent neutron-l3-agent neutron-dhcp-agent \
  neutron-metadata-agent
```

Si modifica il file `/etc/neutron/neutron.conf` nel seguente modo:

```
[database]
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron

[DEFAULT]
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
rpc_backend = rabbit
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS

[nova]
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]`. Si modifica il file `/etc/neutron/plugins/ml2/ml2_conf.ini` nel seguente modo:

```
[ml2]
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = linuxbridge,l2population
extension_drivers = port_security

[ml2_type_flat]
flat_networks = provider

[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
enable_ipset = True
```

Si modifica il file `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` nel seguente modo:

```
[linux_bridge]
physical_interface_mappings = provider:enp0s25

[vxlan]
enable_vxlan = True
local_ip = 130.192.1.90
l2_population = True

[securitygroup]
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Si modifica il file `/etc/neutron/l3_agent.ini` nel seguente modo:

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
external_network_bridge =
```

Si modifica il file `/etc/neutron/dhcp_agent.ini` nel seguente modo:

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

Si modifica il file `/etc/neutron/metadata_agent.ini` nel seguente modo:

```
[DEFAULT]
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

Dove `METADATA_SECRET` è una stringa segreta utilizzata col proxy. Si modifica il file `/etc/nova/nova.conf` nel seguente modo:

```
[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Dove `METADATA_SECRET` è la stessa stringa segreta generata in precedenza. Si popola il database e si riavviano i servizi con:

```
# su -s /bin/sh -c "neutron-db-manage \
  --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
# service nova-api restart
# service neutron-server restart
# service neutron-linuxbridge-agent restart
# service neutron-dhcp-agent restart
# service neutron-metadata-agent restart
# service neutron-l3-agent restart
```

Compute node

Si installa il package con:

```
# apt-get install neutron-linuxbridge-agent
```

Si modifica il file `/etc/neutron/neutron.conf` nel seguente modo:

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = RABBIT_PASS
```

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]` ed eliminando le opzioni connection nella sezione `[database]`. Si modifica il file `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` nel seguente modo:

```
[linux_bridge]
physical_interface_mappings = provider:enp0s25

[vxlan]
enable_vxlan = True
local_ip = 130.192.1.91
l2_population = True

[securitygroup]
enable_security_group = True
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Si modifica il file `/etc/nova/nova.conf` nel seguente modo:

```
[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
```

Si riavviano i servizi con:

```
# service nova-compute restart
# service neutron-linuxbridge-agent restart
```

A.1.9 Dashboard (Horizon)

Questo servizio deve essere installato nel controller node.

Installazione e configurazione package

Si installa il package con:

```
# apt-get install openstack-dashboard
```

Si modifica il file `/etc/openstack-dashboard/local_settings.py` nel seguente modo:

```
OPENSTACK_HOST = "controller"
ALLOWED_HOSTS = ['*', ]
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': 'controller:11211',
    }
}
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 2,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': True,
    'enable_quotas': True,
    'enable_distributed_router': True,
    'enable_ha_router': True,
    'enable_lb': True,
    'enable_firewall': True,
    'enable_vpn': True,
    'enable_fip_topology_check': True,
}
TIME_ZONE = "UTC"
```

Rimuovendo tutte le altre configurazioni della sessione di memorizzazione memcached. Si riavvia il servizio con:

```
# service apache2 reload
```

Adesso la dashboard sarà accessibile all'indirizzo `http://controller/horizon`. Se non fosse possibile visualizzare la finestra di accesso bisogna apportare un'ulteriore modifica e cioè aprire il file di configurazione `/etc/apache2/conf-available/openstack-dashboard.conf` ed aggiungere:

```
WSGIApplicationGroup %{GLOBAL}
```

Infine si riavvia il servizio con:

```
# service apache2 reload
```

A.1.10 Key Manager service (Barbican)

Questo servizio deve essere installato nel controller node.

Creazione database

Si accede come utente root e si crea il database:

```
$ mysql -u root -p

mysql> CREATE DATABASE barbican;
mysql> GRANT ALL PRIVILEGES ON barbican.* TO 'barbican'@'localhost' \
```

```
  IDENTIFIED BY 'BARBICAN_DBPASS';
mysql> GRANT ALL PRIVILEGES ON barbican.* TO 'barbican'@'%' \
  IDENTIFIED BY 'BARBICAN_DBPASS';
```

Creazione endpoint

Si crea il servizio e gli endpoint con:

```
$ . admin-openrc
$ openstack user create --domain default --password-prompt barbican
$ openstack role add --project service --user barbican admin
$ openstack role create creator
$ openstack role add --project service --user barbican creator
$ openstack service create --name barbican --description "Key Manager" \
  key-manager
$ openstack endpoint create --region RegionOne \
  key-manager public http://controller:9311
$ openstack endpoint create --region RegionOne \
  key-manager internal http://controller:9311
$ openstack endpoint create --region RegionOne \
  key-manager admin http://controller:9311
```

Installazione e configurazione package

Si installano i package con:

```
# apt-get install barbican-api barbican-keystone-listener barbican-worker \
  python-barbicanclient
```

Si modifica il file `/etc/barbican/barbican.conf` nel seguente modo:

```
[DEFAULT]
sql_connection = mysql+pymysql://barbican:BARBICAN_DBPASS@controller/barbican
transport_url = rabbit://openstack:RABBIT_PASS@controller
db_auto_create = False

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = barbican
password = BARBICAN_PASS
```

Rimuovendo tutte le altre opzioni della sezione `[keystone_authtoken]`. Si modifica il file `/etc/barbican/barbican-api-paste.ini` nel seguente modo:

```
[composite:main]
use = egg:Paste#urlmap
/: barbican_version
/v1: barbican-api-keystone

[filter:keystone_authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
```

```
auth_plugin = password
username = barbican
password = barbican_password
user_domain_id = USER_DOMAIN_ID
project_name = service
project_domain_id = PROJECT_DOMAIN_ID
auth_uri = http://controller:5000/v3
auth_url = http://controller:35357/v3
auth_version = v3.0
```

Dove `USER_DOMAIN_ID` e `PROJECT_DOMAIN_ID` devono essere sostituiti dai corrispondenti valori visualizzabili eseguendo il comando:

```
$ openstack domain list
```

Si popola il database e si riavviano i servizi con:

```
# su -s /bin/sh -c "barbican-manage db upgrade" barbican
# service apache2 restart
# service barbican-keystone-listener restart
```

A.2 Installazione e configurazione Open Source MANO

Questa guida è estratta dalla documentazione ufficiale Open Source MANO [35]. La versione utilizzata per l'installazione è Open Source MANO TWO, la macchina su cui verrà installato è caratterizzata da CPU Intel core i5 di quinta generazione, 16 GB di RAM e sistema operativo Ubuntu 16.04.2 LTS.

A.2.1 Configurazione LXD

La prima operazione da svolgere è la configurazione della macchina all'utilizzo di LXD. Si aprirà una shell e si eseguiranno i seguenti comandi:

```
$ sudo apt-get update
$ sudo apt-get install zfs
$ sudo apt install lxd
$ newgrp lxd
```

Si inizializzerà LXD rispondendo alle varie richieste come specificato di seguito:

```
$ sudo lxd init
       Name of the storage backend to use (dir or zfs) [default=zfs]: zfs
       Create a new ZFS pool (yes/no) [default=yes]? yes
       Name of the new ZFS pool [default=lxd]: lxdpool
       Would you like to use an existing block device \
         (yes/no) [default=no]? no
       Size in GB of the new loop device (1GB minimum) [default=15]: 15
       Would you like LXD to be available over the network \
         (yes/no) [default=no]? yes
       Do you want to configure the LXD bridge (yes/no) [default=yes]? yes
       Would you like to setup a network bridge for LXD containers now? yes
       Bridge interface name: lxdbr0
       Do you want to setup an IPv4 subnet? Yes
              IPv4 address: 10.54.188.1
              IPv4 CIDR mask: 24
              First DHCP address: 10.54.188.2
```

```
            Last DHCP address: 10.54.188.254
            Max number of DHCP client: 252
            Do you want to NAT the IPv4 traffic? Yes
    Do you want to setup an IPv6 subnet? No
```

A.2.2 Installazione da file binari

Si scaricherà lo script e si avvierà l'installazione:

```
$ wget https://osm-download.etsi.org/ftp/osm-2.0-two/install_osm.sh
$ chmod +x install_osm.sh
$ ./install_osm.sh -b tags/v2.0.2
```

Alla conclusione dell'installazione verrà assegnato un indirizzo IP appartenente alla rete indicata nella fase di inizializzazione ad ogni componente OSM, tali indirizzi saranno visualizzabili col comando:

```
$ lxc list
```

ed in questa specifica installazione gli IP assegnati ai contenitori saranno:

- RO: 10.54.188.42
- VCA: 10.54.188.80
- SO-ub: 10.54.188.202

A.2.3 Controllare l'installazione

Conclusa l'installazione si potrà aprire un browser (è consigliato Google Chrome versione 50 o successivo) e ci si potrà collegare all'indirizzo `https://indirizzo:8443` dove indirizzo corrisponde all'IP assegnato al contenitore SO-ub, verrà visualizzato un avviso associato a problemi di sicurezza perché non è presente un certificato valido per il sito ma sarà possibile procedere e visualizzare la schermata di login, le credenziali saranno:

- Username: admin
- Password: admin

Nel caso di problemi può essere utile verificare che le porte 8443, 8000, 4567, 8008, 80, 9090 siano accessibili.

A.2.4 Aggiunta VIM

Il VIM che verrà aggiunto ad OSM è OpenStack. I passi da seguire sono:

1. Si dovrà garantire che gli endpoint OpenStack siano raggiungibili da OSM, ad esempio si può utilizzare il tool `nmap` per verificare che le porte siano raggiungibili ed aperte:

   ```
   $ nmap -sT -p 5000 controller
   $ nmap -sT -p 35357 controller
   $ nmap -sT -p 9292 controller
   $ nmap -sT -p 8774 controller
   $ nmap -sT -p 9696 controller
   ```

2. Si crea una rete di gestione raggiungibile da OSM attraverso cui il VCA configurerà le VNF, i seguenti comandi devono essere eseguiti in una shell del controller OpenStack:

```
$ neutron net-create mgmt --provider:network_type=flat \
  --provider:physical_network=provider --router:external --shared
$ neutron subnet-create --name subnet-mgmt mgmt 10.208.0.0/24 \
  --allocation-pool start=10.208.0.2,end=10.208.0.100
```

   Per verificare che la rete sia raggiungibile da OSM si accede al VCA e si utilizza il comando ping:

```
$ lxc exec VCA bash
VCA# ping 10.208.0.1
```

3. Si crea un utente con il ruolo di amministratore, accedendo ad una shell nel controller OpenStack si possono digitare i seguenti comandi:

```
$ openstack user create --domain default --password-prompt osm-user
$ openstack role add --project service --user osm-user admin
```

4. Si carica l'immagine che dovrà essere utilizzata dalla VNF in OpenStack eseguendo i seguenti comandi in una shell del controller OpenStack:

```
$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
$ openstack image create --file="./cirros-0.3.4-x86_64-disk.img" \
  --container-format=bare --disk-format=qcow2 --public cirros034
```

5. Si deve aggiungere il permesso all'utilizzo di SSH nel gruppo 'default' quindi si accede ad una shell nel controller OpenStack e si digitano i seguenti comandi:

```
$ openstack security group rule create --proto tcp --dst-port 22 default
```

6. Si configura keystone ad utilizzare le API v2.0 accedendo nel controller al file `/etc/keystone/keystone.conf` ed aggiungendo:

```
[identity]
default_domain_id=DOMAIN_ID
```

   dove `DOMAIN_ID` è l'identificativo del dominio 'default' ottenibile col comando:

```
$ openstack domain list
```

   Infine si sincronizza il database e si riavviano i servizi:

```
# su -s /bin/sh -c "keystone-manage -config-file \
  /etc/keystone/keystone.conf db_sync" keystone
# service apache2 restart
# service keystone restart
```

7. Si aggiunge il data center OpenStack ad OSM con i seguenti comandi:

```
$ lxc exec RO bash
RO# export OPENMANO_TENANT=osm
RO# openmano datacenter-create openstack-site \
     http://130.192.1.90:5000/v2.0 --type openstack \
     --description "OpenStack site"
RO# openmano datacenter-attach openstack-site --user=osm-user \
     --password=USER_PASS --vim-tenant-name=admin
```

   dove `USER_PASS` è la password associata all'utente osm-user di OpenStack.

A.3 Installazione e configurazione Open CIT

Questa guida è estratta dalla documentazione ufficiale Open CIT [42]. L'installazione di Open CIT 3.2 si compone di due fasi: nella prima fase vengono generati i file binari partendo dal codice sorgente, nella seconda fase è possibile installare e configurare i vari componenti Open CIT.

A.3.1 Generazione file binari

Questa operazione deve essere eseguita in una macchina diversa da quella su cui si andrà ad installare Open CIT. Nel nostro caso si utilizzerà una macchina virtuale caratterizzata da 2 CPU, 4 GB di RAM, 100 GB di spazio su disco e sistema operativo Ubuntu 16.04.2 LTS con versione del kernel 4.10.0-27-generic.

Preparazione dell'ambiente

Si installano i package necessari:

```
# apt-get update
# apt-get install ssh ant makeself git make gcc g++ openssl libssl-dev unzip \
  nsis zip -y
```

Si installa la versione Java 1.7.0_51, collegandosi al sito Oracle[1] è possibile scaricare l'archivio `jdk-7u51-linux-x64.tar.gz`, in seguito si possono eseguire i seguenti comandi:

```
# mv jdk-7u51-linux-x64.tar.gz jdk-1.7.0_51-linux-x64.tar.gz
# gzip -dc jdk-1.7.0_51-linux-x64.tar.gz | tar xf -
# mv jdk1.7.0_51 /usr/lib/jvm #Create the jvm directory if it doesn't exist
# cd /etc/alternatives/
# ln -s /usr/lib/jvm/jdk1.7.0_51/bin/java
# ln -s /usr/lib/jvm/jdk1.7.0_51/bin/javac
# cd /usr/bin/
# ln -s /etc/alternatives/java
# ln -s /etc/alternatives/javac
```

Si scarica ed installa Apache Maven versione 3.3.1:

```
# wget http://archive.apache.org/dist/maven/maven-3/3.3.1/binaries/apache-
    maven-3.3.1-bin.zip
# unzip apache-maven-3.3.1-bin.zip
# mv apache-maven-3.3.1 /usr/local
```

Si modifica il file di ambiente ∼`/.bashrc` aggiungendo:

```
JAVA_HOME=/usr/lib/jvm/jdk1.7.0_51
export JAVA_HOME PATH=$PATH:$JAVA_HOME
export MAVEN_HOME=/usr/local/apache-maven-3.3.1
export M2=$MAVEN_HOME/bin
export MAVEN_OPTS="-Xmx1024m -XX:MaxPermSize=1024m" PATH=$PATH:$M2
```

Si riavvia la sessione ed è possibile verificare che tutto sia stato installato correttamente con:

```
# mvn -version
```

[1] http://www.oracle.com/technetwork/java/javase/downloads/index.html

Scaricamento e compilazione codice sorgente

Si accede alla home directory dell'utente root e si scarica il codice sorgente con i seguenti comandi:

```
# cd /root
# git clone https://github.com/opencit/opencit-external-artifacts
# git clone https://github.com/opencit/opencit-contrib
# git clone https://github.com/opencit/opencit-util
# git clone https://github.com/opencit/opencit-privacyca
# git clone https://github.com/opencit/opencit-tboot-xm
# git clone https://github.com/opencit/opencit-trustagent
# git clone https://github.com/opencit/opencit
# git clone https://github.com/opencit/opencit-vrtm
# git clone https://github.com/opencit/opencit-policyagent
# git clone https://github.com/opencit/opencit-docker-proxy
# git clone https://github.com/opencit/opencit-kms
# git clone https://github.com/opencit/opencit-director
# git clone https://github.com/opencit/opencit-openstack-extensions
# git clone https://github.com/opencit/opencit-openstack-controller-extensions
# git clone https://github.com/opencit/opencit-attestation-cache-hub
# git clone https://github.com/opencit/opencit-quickstart
# git clone https://github.com/opencit/opencit-tpm-tools-windows
```

Si scaricano dei package esterni e poi si compila `opencit-external-artifacts`:

```
# cd /root/opencit-external-artifacts
# git checkout v1.0+cit-3.2
# wget --no-check-certificate https://mmonit.com/monit/dist/monit-5.5.tar.gz
# mv monit-5.5.tar.gz monit/monit-5.5-linux-src.tgz
# wget --no-check-certificate https://archive.apache.org/dist/tomcat/tomcat-7/
    v7.0.34/bin/apache-tomcat-7.0.34.tar.gz
# mv apache-tomcat-7.0.34.tar.gz apache-tomcat/apache-tomcat-7.0.34.tar.gz
# wget --no-check-certificate https://sourceforge.net/projects/vijava/files/
    vijava/VI%20Java%20API%205.5%20Beta/vijava55b20130927.zip
# unzip vijava55b20130927.zip
# mv vijava55b20130927.jar vijava/vijava-5.5.jar
# mv jdk-7u51-linux-x64.tar.gz jdk/jdk-1.7.0_51-linux-x64.tar.gz
# wget --no-check-certificate https://sourceforge.net/projects/kmip4j/files/
    KMIP4J-V1.0/kmip4j-bin-1.0.zip
# unzip kmip4j-bin-1.0.zip
# mv jar/kmip4j.jar kmip4j/kmip4j-1.0.jar
# wget --no-check-certificate https://sourceforge.net/projects/ext2fsd/files/
    Ext2fsd/0.62/Ext2Fsd-0.62.exe
# mv Ext2Fsd-0.62.exe ext2fsd/Ext2Fsd-0.62.exe
# wget --no-check-certificate https://sourceforge.net/projects/trousers/files/
    tpm-tools/1.3.8/tpm-tools-1.3.8.tar.gz
# mv tpm-tools-1.3.8.tar.gz tpm-tools/tpm-tools-1.3.8.tar.gz
# rm -rf *.zip
# rm -rf *.jar
# rm -rf XenServer-SDK
# ant
```

Si compilano gli altri package seguendo esattamente l'ordine indicato:

```
# cd /root/opencit-contrib
# git checkout v1.0+cit-3.2
# ant
# cd ..
# cd /root/opencit-util
```

```
# git checkout v1.0+cit-3.2
# ant
# cd ..
# cd /root/opencit-privacyca
# git checkout v3.2
# ant
# cd /root/opencit-tboot-xm
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-trustagent
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit
# git checkout v3.2
# ant
# cd /root/opencit-vrtm
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-policyagent
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-docker-proxy
# git checkout v3.2
# ant
# cd /root/opencit-kms
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-director
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-openstack-extensions
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-openstack-controller-extensions
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-attestation-cache-hub
# git checkout v3.2
# ant
# cd ..
# cd /root/opencit-quickstart
# git checkout v3.2
# ant
```

In alcuni casi la compilazione di `opencit-director` potrebbe provocare degli errori, in questi casi potrebbe essere utile eseguire i seguenti comandi:

```
# cd /root/opencit-director/features/director-swift/
# mvn -DskipTests=true clean install
# cd ../..
```

```
# ant
```

Adesso le varie cartelle conterranno i file binari di installazione dei componenti Open CIT.

A.3.2 Installazione Open CIT

Questa operazione verrà eseguita nella macchina su cui verranno installati la maggior parte dei componenti Open CIT. La macchina sarà caratterizzata da CPU Intel core i5 di quinta generazione, 16 GB di RAM, sistema operativo Ubuntu 16.04.2 LTS ed avrà l'indirizzo IP 130.192.1.110. Inoltre nella procedura di installazione saranno coinvolti anche il controller node ed il compute node che avranno indirizzo IP rispettivamente 130.192.1.90 e 130.192.1.91, saranno equipaggiate di TPM 1.2, della tecnologia Intel TXT ed entrambe dovranno avere la versione del kernel 4.10.0-27-generic. Un vincolo da rispettare è la presenza di un data center con una versione OpenStack non successiva a Mitaka.

L'installazione potrebbe essere eseguita singolarmente per ogni componente Open CIT oppure tramite una procedura automatica, si è scelta la seconda opzione.

Configurazione ambiente

La prima operazione da svolgere è la creazione di un file denominato `adminrc` contenente i dati associati al progetto OpenStack nella cartella `/root` del controller node, il file avrà il seguente contenuto:

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_DEFAULT_DOMAIN=default
```

dove `ADMIN_PASS` deve essere sostituita dalla password dell'utente amministratore. In seguito sarà possibile accedere alla macchina su cui si vogliono installare i componenti Open CIT ed eseguire:

```
# ./cit-quickstart-linux-3.2.1-SNAPSHOT.bin
# ./cit-quickstart-linux-3.2.1-SNAPSHOT-integrity.bin
# ./cit-quickstart-linux-3.2.1-SNAPSHOT-confidentiality.bin
```

Procedura guidata

La procedura guidata è accessibile dal browser all'indirizzo `http://localhost`, sarà possibile configurare alcuni elementi prima di avviare la procedura di installazione. I passi da seguire sono:

1. ambiente: permette di scegliere se installare tutti i componenti Open CIT o appoggiarsi ad un fornitore di servizi cloud già esistente. Nel nostro caso installeremo tutto quindi sceglieremo l'opzione rete privata;

2. caratteristiche: permette di configurare alcune caratteristiche. Sono già preimpostate quelle relative all'attestazione di BIOS, kernel, hypervisor, sistema operativo, applicazioni, driver e configurazioni. Le nostre scelte sono relative alla protezione del carico di lavoro per cui si imposta sia l'integrità sia la cifratura delle macchine virtuali eseguita utilizzando OpenStack Barbican, e all'integrazione automatica di Open CIT con OpenStack Nova, Horizon e Glance;

3. schema: permette di scegliere se installare tutti i componenti Open CIT in un unico host o se scegliere dove installare ogni singolo componente. Nel nostro caso sceglieremo l'host su cui installare ogni componente, inoltre specificheremo che sono presenti già delle configurazioni esistenti di OpenStack Glance e Barbican;

4. impostazioni: si dovranno configurare i dettagli degli elementi già esistenti secondo il seguente schema:

```
Barbican project id: "BARBICAN_PROJECT_ID"
Barbican server url: "http://controller:9311"
Keystone server url: "http://controller :5000"
Username: "barbican"
Password: "BARBICAN_PASS"
Barbican tenant name: "service"

Glance server url: "http://controller:9292"
Keystone server url: "http://controller :5000"
Username: "glance"
Password: "GLANCE_PASS"
Openstack tenant name: "service"
```

   dove `BARBICAN_PROJECT_ID` è l'identificativo associato al progetto e il suo valore può essere visualizzato nel controller node col comando

```
$ openstack service list
```

5. credenziali: permette di impostare per ogni componente l'indirizzo IP o l'hostname della macchina su cui dovrà essere installato. Si dovrà permettere l'accesso ad ognuna di esse come utente root tramite SSH indicando in questa procedura la password di accesso. Nel nostro caso sono state utilizzate le seguenti configurazioni:

   - 130.192.1.110 per Attestation Service, Trust Director, Attestation Reporting Hub, Key Broker, Key Broker proxy;
   - controller per Trust Agent ed OpenStack extensions;
   - compute1 per Trust Agent.

Ad installazione completata verranno elencati tutti i servizi con l'URL a cui sono raggiungibili e con le credenziali di accesso.

Per completare la configurazione si deve accedere all'Attestation Server attraverso la sua interfaccia grafica, si va sul tab Administration, si clicca su Pending Requests e ci dovrebbe essere una richiesta del KMS proxy, si clicca il pulsante Details, si impostano i ruoli Attestation e Challenger ed infine si approva la richiesta.

Un'altra operazione da eseguire è l'importazione dei certificati dell'Attestation Server nel KMS, in particolare si accede nell'Attestation Server, si va nel tab Administration, si clicca su View Certificates e si scaricano i certificati cliccando su Download Configuration data bundle, in seguito si accede nel Key Broker, si va sul tab Settings, si clicca su More Settings e poi su Upload Settings, si seleziona il data bundle precedentemente scaricato e si esegue il caricamento.

OpenStack extensions

La procedura di installazione delle OpenStack extensions dovrebbe essere automatica ma quando è necessario installarle in una macchina differente da quella in cui si esegue la procedura guidata si possono verificare dei problemi. In questo caso è consigliabile scaricare direttamente da GitHub le OpenStack extensions[2] per la versione Mitaka ed installarle. L'installazione deve avvenire nel

[2] https://github.com/opencit/opencit-openstack-extensions

controller node, l'archivio scaricato verrà estratto nella directory `/root`, si accederà all'interno di esso dove sarà presente il file `setup.cfg` che verrà così modificato:

```
CIT_IP=130.192.1.110
CIT_PORT=8443
CIT_AUTH_BLOB=TAG_USER:TAG_PASS
```

dove `TAG_USER` e `TAG_PASS` rappresentano il nome utente e la password dell'utente tagadmin presente nell'Attestation Server. Per conoscere queste credenziali è necessario accedere al file `mtwilson.env` nella macchina su cui è installato l'Attestation Server. Infine sarà possibile avviare l'installazione delle OpenStack extensions con:

```
# ./setup install
```

Installazione Trust Agent

L'installazione del Trust Agent sia sul controller node sia sul compute node non avviene automaticamente ma deve essere eseguita manualmente, essa si compone dei seguenti passi:

1. si accede al BIOS, sotto System configuration menu si accede a Device configurations e si disabilita Trusted Execution Technology (TXT). Si accede a Security menu, sotto User tools si trova TPM embedded security, si seleziona TPM reset to factory default e si conferma la scelta. Adesso è possibile salvare e riavviare il sistema, al riavvio si confermerà la cancellazione del TPM;
2. si scarica il `SINIT ACM` dal sito Intel[3] in base alla propria CPU ed in particolare nel nostro caso sarà necessario il modulo per CPU di terza generazione denominato `3rd_gen_i5_i7_SINIT_67.zip`, si estrae l'archivio che conterrà un file binario e si copierà nella cartella `/boot`;
3. nella directory `/root` sarà presente il file binario per l'installazione del Trust Agent e sarà possibile eseguirlo con:

   ```
   # ./cit3-openstack-trusted-node-ubuntu.bin
   ```

   durante l'installazione viene aggiunta una nuova modalità di avvio denominata `tboot`. Al termine dell'installazione si può riavviare il sistema;
4. si accede nel BIOS, poi si accede a Security menu, sotto User tools si trova TPM embedded security dove si dovranno impostare le funzionalità:

   - TPM device available;
   - Embedded security device state;
   - Reset authentication credential impostato su yes.

   si salva tutto e si riavvia il pc che si spegnerà ed accenderà un paio di volte;
5. si accede nuovamente al BIOS e si abilita TXT, si salva tutto e si riavvia in modalità `tboot`;
6. si accede alla cartella `/root` e si avvia nuovamente il file binario per l'installazione del Trust Agent con:

   ```
   # ./cit3-openstack-trusted-node-ubuntu.bin
   ```

   durante l'installazione viene aggiunta una nuova modalità di avvio denominata `TCB-Protection`. Al termine dell'installazione si può configurare questa nuova modalità come quella di avvio predefinita verificando che il file `/etc/grub.d/40-custom` contenga effettivamente i dati per avviare il sistema in modalità `TCB-Protection`. In questo caso si può rinominare il file ed aggiornare grub con i comandi:

[3] https://software.intel.com/en-us/articles/intel-trusted-execution-technology/

```
# mv /etc/grub.d/40-custom /etc/grub.d/01-cit
# update-grub
```

infine si riavvia il sistema in modalità `TCB-Protection`.

Un problema che si potrebbe notare quando il Trust Agent deve inviare le misure all'Attestation Server è la mancanza del file `aikquote`, in questo caso è necessario prelevare il file dalla macchina in cui sono stati compilati i sorgenti Open CIT nella cartella `/root/opencit-contrib/features/tpm-agent-tools/target` e copiarlo nella macchina su cui è installato il Trust Agent nella cartella `/opt/trustagent/share/tpmquote/bin/`.

A.4 Configurazione firewall

La configurazione del firewall è un requisito fondamentale per esporre i servizi installati solo a chi è realmente autorizzato ad utilizzarli. La nostra configurazione utilizza degli indirizzi IP pubblici quindi è necessario filtrare tutte le richieste che arrivano da indirizzi non autorizzati. Su ogni macchina utilizzata per implementare la soluzione sono stati analizzati i servizi attivi, le relative porte che espongono e quale macchina potrebbe richiedere l'accesso a tali servizi, così è stato configurato il firewall.

A.4.1 Controller node

Il controller node implementa i principali servizi OpenStack ed il Trust Agent. La tabella A.1 illustra i servizi attivi, la porta associata e il protocollo utilizzato. Sono mostrati solo i servizi per

Servizio	*Porta*	*Protocollo*
SSH	22	Tcp
Chrony	123	Udp
Mariadb	3306	Tcp
Rabbitmq-base	5672	Tcp
Rabbitmq-epmd	4369	Tcp
Rabbitmq-beam	25672	Tcp
Memcached	11211	Tcp/Udp
Keystone	5000	Tcp
Keystone	35357	Tcp
Glance	9191	Tcp
Glance	9292	Tcp
Nova	6080	Tcp
Nova	8774	Tcp
Nova	8775	Tcp
Neutron	9696	Tcp
Barbican	9311	Tcp
Barbican	9312	Tcp
Trust Agent	1443	Tcp

Tabella A.1. Servizi attivi nel controller node.

cui è richiesto l'accesso anche da macchine esterne rispetto a quella su cui sono installati.

A.4.2 Compute node

Il compute node implementa il servizio OpenStack Nova ed il Trust Agent. La tabella A.2 illustra i servizi attivi, il protocollo utilizzato e la porta associata al servizio. Sono mostrati solo i servizi per cui è richiesto l'accesso anche da macchine esterne rispetto a quella su cui sono installati.

Servizio	*Porta*	*Protocollo*
SSH	22	Tcp
Nova	5900:5999	Tcp
Trust Agent	1443	Tcp

Tabella A.2. Servizi attivi nel compute node.

A.4.3 NUC

Il NUC implementa i servizi Open Source MANO ed i principali servizi Open CIT. La tabella A.3 illustra i servizi attivi, il protocollo utilizzato e la porta associata al servizio. Sono mostrati solo i

Servizio	*Porta*	*Protocollo*
Attestation Service	8443	Tcp
Key Broker proxy	21080	Tcp

Tabella A.3. Servizi attivi nel NUC.

servizi per cui è richiesto l'accesso anche da macchine esterne rispetto a quella su cui sono installati quindi non sono presenti i servizi Open Source MANO né il Trust Director, l'Attestation Hub o il Key Broker perché sono accessibili solo localmente.

A.4.4 Regole configurate

Il firewall è stato configurato utilizzando `iptables`. Su ogni macchina sono state aggiunte le regole per consentire solo alle macchine che realmente lo richiedono l'accesso ai servizi specifici ed alla fine su ogni macchina è stata aggiunta la regola per scartare tutti gli altri pacchetti. Le configurazioni realizzate sono le seguenti:

- Controller:

```
# iptables -A INPUT -s 130.192.1.91 -d 130.192.1.90 -p tcp \
  --match multiport --dport 3306,5672,4369,25672,11211,5000,\
  35357,9191,9292,6080,8774,8775,9696,9311,9312 -j ACCEPT
# iptables -A INPUT -s 130.192.1.91 -d 130.192.1.90 -p udp \
  --match multiport --dport 123,11211 -j ACCEPT
# iptables -A INPUT -s 130.192.1.110 -d 130.192.1.90 -p tcp \
  --match multiport --dport 3306,5672,4369,25672,11211,5000,\
  35357,9191,9292,6080,8774,8775,9696,9311,9312 -j ACCEPT
# iptables -A INPUT -s 130.192.1.110 -d 130.192.1.90 -p udp \
  --match multiport --dport 123,11211 -j ACCEPT
# iptables -A INPUT -s 130.192.1.110 -d 130.192.1.90 -p tcp \
  --dport 1443 -j ACCEPT
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -j DROP
```

  Così si consente al compute node ed al NUC su cui è installato Open Source MANO l'accesso a tutti i servizi OpenStack. Si consente ai componenti Open CIT del NUC di accedere al Trust Agent e si scartano tutti gli altri pacchetti. Si noti che non viene resa persistente la regola che consente al Trust Director di accedere tramite SSH al Trust Agent per inserire la Trust policy non virtualizzata, siccome questo è un evento che si verifica raramente in questi casi sarà necessario aggiungere temporaneamente la seguente regola:

```
# iptables -I INPUT -s 130.192.1.110 -d 130.192.1.90 -p tcp \
  --dport 22 -j ACCEPT
```

- Compute:

```
# iptables -A INPUT -s 130.192.1.90 -d 130.192.1.91 -p tcp \
  --dport 5900:5999 -j ACCEPT
# iptables -A INPUT -s 130.192.1.110 -d 130.192.1.91 -p tcp \
  --dport 1443 -j ACCEPT
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -j DROP
```

  Così si consente al controller node l'accesso alle istanze gestite da Nova. Si consente ai componenti Open CIT del NUC di accedere al Trust Agent e si scartano tutti gli altri pacchetti. Si noti che non viene resa persistente la regola che consente al Trust Director di accedere tramite SSH al Trust Agent per inserire la Trust policy non virtualizzata, siccome questo è un evento che si verifica raramente in questi casi sarà necessario aggiungere temporaneamente la seguente regola:

```
# iptables -I INPUT -s 130.192.1.110 -d 130.192.1.91 -p tcp \
  --dport 22 -j ACCEPT
```

- NUC:

```
# iptables -A INPUT -s 130.192.1.90 -d 130.192.1.110 -p tcp \
  --dport 8443 -j ACCEPT
# iptables -A INPUT -s 130.192.1.91 -d 130.192.1.110 -p tcp \
  --dport 8443 -j ACCEPT
# iptables -A INPUT -s 130.192.1.90 -d 130.192.1.110 -p tcp \
  --dport 21080 -j ACCEPT
# iptables -A INPUT -s 130.192.1.91 -d 130.192.1.110 -p tcp \
  --dport 21080 -j ACCEPT
# iptables -A INPUT -i lo -j ACCEPT
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A INPUT -j DROP
```

  Così si consente al Trust Agent del controller node e del compute node di comunicare con l'Attestation Server e con il KMS proxy e si scartano tutti gli altri pacchetti.

Bisogna notare che non basta modificare le regole del firewall per renderle persistenti infatti ad ogni riavvio tali modifiche non saranno più presenti quindi dopo aver apportato le modifiche si potrà utilizzare il pacchetto `iptables-persistent` per rendere le regole persistenti. Esso si potrà installare con:

```
# apt install iptables-persistent
```

e dopo aver apportato nuove modifiche al firewall si potranno salvare sia le regole per i pacchetti IPv4 sia quelle per i pacchetti IPv6 con:

```
# iptables-save > /etc/iptables/rules.v4
# iptables-save > /etc/iptables/rules.v6
```

A.5 Soluzione proposta

Questa sezione spiega come è possibile ricreare l'architettura illustrata nella tesi ed applicare la soluzione proposta.

La prima operazione da svolgere per ricreare la stessa struttura architetturale utilizzata nella tesi è l'installazione dei software. Si consiglia di installare prima OpenStack, poi Open Source MANO ed infine Open CIT. Le procedure di installazione sono illustrate nelle Sezioni A.1, A.2 e A.3.

A seguire sarà necessario personalizzare il file `vimconn_openstack.py` allegato a questo elaborato inserendo i dati con cui collegarsi all'Attestation Server cioè si dovranno modificare le seguenti righe:

```
ATTESTATION_SERVICE = {
    'IP': 'INDIRIZZO_IP',
    'port': 'PORTA',
    'auth_blob': 'TAG_USER:TAG_PASSWORD',
    'auth_blob_vm': 'USER:PASSWORD'
}
```

`INDIRIZZO_IP` corrisponde al valore dell'indirizzo IP a cui è raggiungibile l'Attestation Server, `PORTA` rappresenta la porta su cui è in ascolto l'Attestation Server, `TAG_USER` e `TAG_PASSWORD` sono le credenziali dell'utente che può gestire gli asset tag nell'Attestation Server, `USER` e `PASSWORD` sono le credenziali dell'utente amministratore dell'Attestation Server. Sarà necessario anche aggiungere le informazioni sul controller node e sul compute node utilizzati. Eventualmente è possibile aggiungere anche le informazioni di altri nodi che è necessario attestare prima di avviare un nuovo servizio di rete, per ogni nodo dovrà essere inserito un nome che lo identifica nel file, il suo hostname e l'indirizzo IP come mostrato nel codice seguente:

```
REGISTERED_HOSTS = {
    'NOME_NODO': {
        'hostname': 'HOSTNAME',
        'ip': 'INDIRIZZO_IP'
    }
}
```

Terminate le modifiche al file sarà necessario aprire una shell nella macchina su cui è installato Open Source MANO. Come prima operazione si consiglia di accedere al contenitore RO e rinominare il file che dovrà essere rimpiazzato con i seguenti comandi:

```
$ lxc exec RO bash

RO# mv /usr/lib/python2.7/dist-packages/osm_ro/vimconn_openstack.py \
    /usr/lib/python2.7/dist-packages/osm_ro/vimconn_openstack.bak
```

Si potrà aggiungere il file modificato e si riavvierà il contenitore con:

```
$ lxc file push vimconn_openstack.py \
  RO/usr/lib/python2.7/dist-packages/osm_ro/
$ lxc restart RO
```

Sarà necessario aggiungere il certificato digitale dell'Attestation Server all'interno del RO per creare una connessione sicura fra i due elementi. Il certificato potrà essere prelevato dalla macchina su cui è installato l'Attestation Server al percorso `/etc/intel/cloudsecurity/ssl.crt` e potrà essere copiato nel RO col comando:

```
$ lxc file push ssl.crt RO/usr/local/share/ca-certificates/
```

Se si volesse usare un'altra cartella per mantenere il certificato sarà necessario modificare il percorso all'interno del codice sorgente.

Adesso, per ogni servizio di rete che si dovrà avviare, ci sarà una verifica preventiva dell'attestazione sia dei nodi fisici sia delle macchine virtuali.

Appendice B

Manuale dello sviluppatore

In questo manuale presenteremo le principali API REST esposte dall'Attestation Server nella Sezione B.1.

B.1 API Attestation Server

Le API sono estratte dalla documentazione ufficiale Open CIT [42]. A seguire vengono descritte le principali API esposte dall'Attestation Server. Tutte saranno utilizzabili contattando l'Attestation Server alla porta 8443 ed autenticandosi.

Gestione certificati

- `GET /mtwilson/v2/ca-certificates/TYPE_CERT`
  - parametri: tipo di certificato richiesto infatti `TYPE_CERT` potrà assumere i valori `root`, `saml`, `tls`, `privacy`;
  - dati restituiti: certificato X.509 in formato `PEM`;
  - descrizione: recupera i dettagli del certificato specificato.

Gestione AIK

- `POST /mtwilson/v2/hosts/HOST_UUID/aik-certificates`
  - parametri: certificato AIK; `HOST_UUID` deve essere sostituito dall'identificativo dell'host su cui creare il certificato;
  - dati restituiti: certificato creato nel sistema, il suo identificativo e l'identificativo dell'host su cui è stato creato; il formato potrà essere `JSON`, `XML` o `YAML`;
  - descrizione: associa il certificato AIK all'host specificato.
- `GET /mtwilson/v2/hosts/HOST_UUID/aik-certificates`
  - parametri: `HOST_UUID` deve essere sostituito dall'identificativo dell'host su cui cercare il certificato;
  - dati restituiti: lista di certificati AIK associati all'host; il formato dei certificati sarà `PEM`;
  - descrizione: restituisce un unico certificato AIK associato all'host perché attualmente il sistema supporta solo un certificato per host.
- `GET /mtwilson/v2/hosts/HOST_UUID/aiks`

  - parametri: `HOST_UUID` deve essere sostituito dall'identificativo dell'host su cui cercare la chiave AIK;
  - dati restituiti: lista di chiavi AIK presenti sull'host; il formato sarà `JSON`, `XML` o `YAML`;
  - descrizione: restituisce un'unica chiave pubblica AIK e lo SHA1 calcolato su di essa perché attualmente il sistema supporta solo una chiave AIK per host.

Gestione Host

- `POST /mtwilson/v2/hosts`
  - parametri: dettagli sull'host che deve essere registrato cioè l'indirizzo IP, l'URL a cui è raggiungibile il Trust Agent, l'identificativo di BIOS MLE, VMM MLE e della policy TLS;
  - dati restituiti: l'identificativo dell'host appena registrato e la conferma dei dati che erano stati inviati; il formato può essere `JSON`, `XML` o `YAML`;
  - descrizione: registra nel sistema l'host specificato.
- `DELETE /mtwilson/v2/hosts/HOST_UUID`
  - parametri: `HOST_UUID` deve essere sostituito dall'identificativo dell'host da cancellare;
  - dati restituiti: nessuno;
  - descrizione: cancella l'host con l'identificativo specificato.
- `PUT /mtwilson/v2/hosts/HOST_UUID`
  - parametri: gli attributi da aggiornare; `HOST_UUID` deve essere sostituito dall'identificativo dell'host per cui devono essere modificati dei dati;
  - dati restituiti: i dati che sono stati aggiornati, il formato può essere `JSON`, `XML` o `YAML`;
  - descrizione: aggiorna l'host con gli attributi specificati.
- `GET /mtwilson/v2/hosts/HOST_UUID`
  - parametri: `HOST_UUID` deve essere sostituito dall'identificativo dell'host;
  - dati restituiti: dettagli relativi all'host in formato `JSON`, `XML` o `YAML`;
  - descrizione: restituisce i dettagli relativi all'host specificato.
- `GET /mtwilson/v2/hosts?CRITERI`
  - parametri: `CRITERI` deve essere sostituito da criteri che permettono di raffinare la ricerca, verranno inseriti nella forma `NOME=VALORE` e potranno essere concatenati più criteri separandoli col simbolo `&`; fra i criteri che è possibile inserire ci sono:
    `id`: l'identificativo dell'host;
    `nameEqualTo`: l'hostname o l'indirizzo IP dell'host;
    `nameContains`: una sequenza che è possibile trovare nell'hostname o nell'indirizzo IP dell'host;
    `filter`: impostato a falso restituisce tutti gli host presenti senza utilizzare nessun filtro.
  - dati restituiti: lista di host in formato `JSON`, `XML` o `YAML`;
  - descrizione: restituisce la lista di host che corrispondono ai criteri specificati.
- `GET /mtwilson/v2/host-uuids?CRITERI`
  - parametri: `CRITERI` deve essere sostituito da criteri che permettono di raffinare la ricerca, verranno inseriti nella forma `NOME=VALORE` e potranno essere concatenati più criteri separandoli col simbolo `&`; attualmente l'unico criterio che è possibile inserire è:
    `hostNameEqualTo`: l'hostname o l'indirizzo IP dell'host.
  - dati restituiti: lista di identificativi degli host in formato `JSON`, `XML` o `YAML`;

  - descrizione: restituisce la lista di identificativi legati all'hardware degli host specificati con i criteri.

- `POST /mtwilson/v2/rpc/register-hosts`
  - parametri: lista di host da registrare;
  - dati restituiti: lista di host creati e il loro stato; il formato può essere `JSON`, `XML` o `YAML`;
  - descrizione: registra nel sistema la lista di host specificati e per ognuno aggiorna le misure di riferimento.

Gestione attestazione host

- `POST /mtwilson/v2/host-attestations`
  - parametri: identificativo dell'host su cui eseguire l'attestazione;
  - dati restituiti: Trust report in formato `JSON` o `SAML`;
  - descrizione: forza un nuovo ciclo di attestazione per l'host specificato nel parametro.

- `DELETE /mtwilson/v2/host-attestations/ATTESTATION_ID`
  - parametri: `ATTESTATION_ID` deve essere sostituito dall'identificativo dell'attestazione da cancellare;
  - dati restituiti: nessuno;
  - descrizione: cancella l'attestazione specificata.

- `GET /mtwilson/v2/host-attestations/ATTESTATION_ID`
  - parametri: `ATTESTATION_ID` deve essere sostituito dall'identificativo dell'attestazione da recuperare;
  - dati restituiti: Trust report in formato `JSON`, `XML` o `YAML`;
  - descrizione: restituisce il Trust report dell'attestazione specificata.

- `GET /mtwilson/v2/host-attestations?CRITERI`
  - parametri: `CRITERI` deve essere sostituito da criteri che permettono di raffinare la ricerca, verranno inseriti nella forma `NOME=VALORE` e potranno essere concatenati più criteri separandoli col simbolo `&`; fra i criteri che è possibile inserire ci sono:
    `id`: l'identificativo dell'attestazione;
    `host_id`: l'identificativo dell'host;
    `nameEqualTo`: l'hostname o l'indirizzo IP dell'host;
    `aik_public_key_sha1`: lo SHA1 calcolato sulla chiave pubblica AIK dell'host;
    `numberOfDays`: il numero di giorni dalla data attuale per cui si devono recuperare le attestazioni;
    `fromDate`: la data iniziale da cui cominciare a recuperare le attestazioni;
    `toDate`: la data finale entro cui recuperare le attestazioni;
    `limit`: il numero massimo di record che possono essere restituiti;
    `filter`: impostato a falso restituisce tutte le attestazioni presenti senza utilizzare nessun filtro.
  - dati restituiti: lista di attestazioni in formato `JSON`, `XML`, `YAML` o `SAML`;
  - descrizione: restituisce la lista di attestazioni secondo i criteri specificati, se il formato specificato per i dati di ritorno è `SAML` allora verrà restituita solo l'ultima attestazione valida che corrisponde ai criteri di ricerca.

Gestione Trust policy

- `POST /mtwilson/v2/trustpolicy-signature`
  - parametri: Trust policy in formato `XML`;
  - dati restituiti: Trust policy firmata in formato `XML`;
  - descrizione: riceve la Trust policy generata dal Trust Director ed appone la sua firma digitale.

Gestione attestazione VM

- `POST /mtwilson/v2/vm-attestations`
  - parametri: identificativo della VM ed hostname o indirizzo IP dell'host su cui si trova la VM;
  - dati restituiti: Trust report in formato `JSON` o `XML`;
  - descrizione: forza un nuovo ciclo di attestazione per la VM specificata nel parametro.
- `DELETE /mtwilson/v2/vm-attestations/ATTESTATION_ID`
  - parametri: `ATTESTATION_ID` deve essere sostituito dall'identificativo dell'attestazione della VM da cancellare;
  - dati restituiti: nessuno;
  - descrizione: cancella l'attestazione specificata.
- `GET /mtwilson/v2/vm-attestations/ATTESTATION_ID`
  - parametri: `ATTESTATION_ID` deve essere sostituito dall'identificativo dell'attestazione della VM da recuperare;
  - dati restituiti: Trust report in formato `JSON`, `XML` o `YAML`;
  - descrizione: restituisce il Trust report dell'attestazione della VM specificata.
- `GET /mtwilson/v2/vm-attestations?CRITERI`
  - parametri: `CRITERI` deve essere sostituito da criteri che permettono di raffinare la ricerca, verranno inseriti nella forma `NOME=VALORE` e potranno essere concatenati più criteri separandoli col simbolo `&`; fra i criteri che è possibile inserire ci sono:
    `vmAttestationIdEqualTo`: l'identificativo dell'attestazione della VM;
    `vmInstanceIdEqualTo`: l'identificativo della VM;
    `hostNameEqualTo`: l'hostname o l'indirizzo IP dell'host;
    `numberOfDays`: il numero di giorni dalla data attuale per cui si devono recuperare le attestazioni;
    `fromDate`: la data iniziale da cui cominciare a recuperare le attestazioni;
    `toDate`: la data finale entro cui recuperare le attestazioni;
    `limit`: il numero massimo di record che possono essere restituiti.
  - dati restituiti: lista di attestazioni VM in formato `JSON`, `XML` o `YAML`;
  - descrizione: restituisce la lista di attestazioni delle VM secondo i criteri specificati.