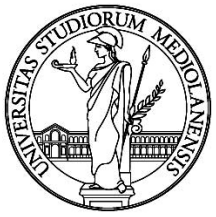



**UNIVERSITÀ DEGLI STUDI DI MILANO**

DIPARTIMENTO DI INFORMATICA

*Corso di Laurea Magistrale in Sicurezza Informatica*


# Valutazione automatica e continua della compliance in ambienti cloud: Il caso di studio FedRAMP


RELATORE

Prof. Claudio Agostino Ardagna

CORRELATORE

Dott. Marco Anisetti

SECONDO CORRELATORE

Prof. Ernesto Damiani

TESI DI LAUREA DI

Patrizio Tufarolo

Matr. 875041



Anno Accademico 2016/2017

*Ai miei genitori e a mio fratello*

# Ringraziamenti

Ringrazio tutti coloro che mi sono stati vicini in questi cinque anni, credendo in me, supportandomi e spronandomi a fare sempre di meglio.

Ringrazio tutti i membri del SESAR Lab, che mi hanno dato la possibilità di apportare un contributo alle attività di ricerca, in particolar modo relative ai progetti CUMULUS e Moon Cloud. Ringrazio in particolare **Filippo Gaudenzi**, supervisore delle mie attività nel suddetto laboratorio con cui ho maturato un rapporto di amicizia e collaborazione.

Un ulteriore ringraziamento va alla prof. Sara Foresti, che ha accettato l'impegno di essere controrelatore di questo lavoro di tesi.

*Patrizio Tufarolo*

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Introduzione

Negli ultimi anni, l'adozione del paradigma cloud ha permesso alle organizzazioni di usufruire di vantaggi prestazionali ed economici nell'erogazione dei servizi IT, grazie sia alle caratteristiche di scalabilità ed elasticità proprie dello stesso, che alla possibilità di allocare risorse in modalità *on-demand*.

Tuttavia, la natura distribuita ed automatizzata della cloud introduce numerose problematiche di sicurezza e valutazione del rischio, soprattutto per quelle realtà in procinto di effettuare migrazioni parziali o totali delle loro infrastrutture tradizionali *on premises*.

La centralizzazione degli aspetti di sicurezza nelle mani di un *cloud service provider* rappresenta infatti uno dei limiti maggiori dell'approccio, e richiede un rapporto di fiducia reciproca tra il fornitore del servizio e il cliente. Questo rapporto di fiducia, attualmente basato sulla reputazione del provider, rappresenta la base per minimizzare il rischio e limitare il perimetro di attacco, e per stabilire le responsabilità di ogni entità coinvolta nella gestione degli incidenti.

Questo lavoro di tesi si pone all'interno della filiera di ricerca sulla *security assurance*. La security assurance mira ad incrementare il livello di attendibilità di un sistema verificandone i comportamenti attesi in caso di fallimento o attacchi. Una delle tecniche che possono essere utilizzate consiste nella produzione di evidenze fidate e replicabili, basate su attività di testing e monitoraggio.

In particolare, il lavoro di tesi ha come obiettivo la verifica continua e automatica della compliance allo standard FedRAMP, il programma governativo americano per la valutazione del rischio e l'autorizzazione all'utilizzo di servizi cloud nelle agenzie federali, che è stato adottato in diversi domini ad alta criticità come ad esempio Amazon AWS e il Dipartimento della Difesa americano.

Ci si è concentrati quindi da un lato sull'analisi e sullo studio dello standard FedRAMP, allo scopo di identificare i controlli di sicurezza (descritti nel documento NIST SP 800-53) di interesse per una valutazione di compliance; dall'altro nell'implementazione dei controlli di sicurezza identificati e nella loro integrazione all'interno di Moon Cloud, una piattaforma a micro-servizi per la trasparency, l'assessment e il monitoraggio continuativo di proprietà non funzionali.

Il lavoro di tesi vuole fornire un'ambiente per la valutazione di compliance continua e automatica allo standard FedRAMP e assumere un ruolo di supporto per tutti gli attori coinvolti nel processo di autorizzazione, in particolare i fornitori di servizi che vogliano attestarne la *readiness*.

Nell'ambito della tesi è stato inoltre redatto un articolo dal titolo "A security benchmark for OpenStack" che, partendo dal benchmark CIS, identifica alcuni controlli di sicurezza specifici per il prodotto in oggetto. Questo è stato sotto-

messo ed accettato alla conferenza IEEE Cloud 2017 in programma dal 25 al 30 Giugno ad Honolulu (Hawaii, USA).

Il lavoro di tesi può essere riassunto come segue:

- *Analisi di FedRAMP*, individuazione dei punti chiave del programma e studio di metodologie a supporto delle attività e degli attori coinvolti nel processo di autorizzazione.
- *Design e implementazione di driver* per la valutazione automatica e continua dei controlli di sicurezza. Al fine di garantire la copertura delle specifiche del framework, sono stati utilizzati due diversi approcci per l'esecuzione dei controlli di sicurezza:
  - *driver per i controlli automatici*, la cui esecuzione avviene in modo autonomo, per l'*assessment* delle proprietà non-funzionali effettivamente implementate nei sistemi informatici;
  - *driver per i controlli ad interazione umana*, effettuati tramite la somministrazione online di questionari, per l'analisi dei processi di business.
- *Integrazione dei controlli di sicurezza automatici* all'interno della piattaforma multi-layer Moon Cloud.
- *Validazione* della soluzione proposta e dei corrispondenti controlli di sicurezza nell'ambito della verifica della compliance della piattaforma Moon Cloud allo standard FedRAMP. La fase di validazione ha anche considerato i costi e l'effort di deployment dei controlli di sicurezza.

L'organizzazione dei capitoli è la seguente:

- **Capitolo 1 - Security assurance: lo stato dell'arte e la sfida** Nel primo capitolo sono analizzate le problematiche di sicurezza nella migrazione di infrastrutture tradizionali verso la *cloud* illustrando le soluzioni proposte dalla letteratura. Tra queste, vengono approfondite le tecnche di *assurance* mediante certificazione e controllo della *compliance* tramite la raccolta di evidenze.
- **Capitolo 2 - Moon Cloud, un framework per il monitoraggio e la security assurance** Nel secondo capitolo viene illustrato Moon Cloud, una soluzione software prodotta dal laboratorio SESAR dell'Università degli Studi di Milano, avente come obiettivo la security assurance.
- **Capitolo 3 - FedRAMP - Federal Risk and Authorization Management Program** Nel terzo capitolo viene analizzato FedRAMP, analizzando i punti chiave del programma, i ruoli e le responsabilità degli attori coinvolti, e identificando i possibili approcci per implementarne i controlli di sicurezza all'interno di Moon Cloud, al fine di fornire uno strumento a supporto delle attività di ciascun attore.
- **Capitolo 4 - Implementazione dei controlli di sicurezza FedRAMP in Moon Cloud** Nel quarto capitolo viene proposto un approccio ibrido orchestrato

tramite l'integrazione di attività di *security assessment* automatiche e ad interazione umana (per i controlli di sicurezza di carattere proceduale e l'analisi dei processi di business) e ne viene illustrata una possibile implementazione. Nel primo caso, viene presentato il framework OpenSCAP, un ecosistema realizzato da Red Hat per le attività di auditing, e ne viene proposta un'integrazione con Moon Cloud. Nel secondo caso, viene presentato un driver Moon Cloud ad interazione umana, che invia questionari personalizzabili tramite template agli utenti interessati, richiedendone la compilazione e creando un report in PDF.

- **Capitolo 5 - Validazione del framework** Nel quinto capitolo viene effettuata la validazione del lavoro svolto, in particolar modo per l'approccio automatico. Il driver OpenSCAP viene eseguito sul deployment in produzione dello stesso Moon Cloud, per effettuarne l'analisi della sicurezza. Come illustrato nel capitolo 2, Moon Cloud è organizzato in microservizi gestiti tramite *container*. In particolare, il deployment è effettuato su una IaaS OpenStack e una PaaS Docker: viene perciò analizzata l'infrastruttura OpenStack sottostante, mediante OpenSCAP e controlli di sicurezza specifici mappati sui requisiti FedRAMP; viene quindi eseguito l'*assessment* dei container dei vari componenti di Moon Cloud e dei canali di comunicazione. Infine, viene proposta una valutazione delle performance del lavoro svolto e saranno trattate alcune delle problematiche riscontrate.

- **Capitolo 6 - Conclusione e sviluppi futuri** Nel capitolo finale vengono mostrati alcuni sviluppi futuri del progetto in questione, che spaziano dall'integrazione dei driver sviluppati con altri standard, per arrivare all'utilizzo dell'approccio illustrato in casistiche analoghe al processo di auditing della compliance.

# Capitolo 1

# Security assurance: lo stato dell'arte e la sfida

## 1.1 Introduzione

In questo capitolo si approfondirà lo stato dell'arte in materia di **security assurance** e **controllo della compliance**, ovvero la verifica della conformità di un'infrastruttura informatica tradizionale, ibrida o cloud, rispetto a una politica, che può essere sviluppata internamente oppure derivata da un più complesso apparato normativo o da uno standard. In particolare verranno trattate le problematiche di sicurezza introdotte dall'adozione di un approccio *cloud*, all'interno dei processi *IT* di un'organizzazione strutturata sulla base di un'infrastruttura informatica tradizionale.

Spesso si fa coincidere il concetto di cloud computing con quello di outsourcing, di fatto presupponendo che l'adozione di tecnologie *cloud* corrisponda all'attitudine di concedere a terzi gli oneri di gestione di una parte dell'infrastruttura informatica. La definizione di *cloud computing* a cui si fa riferimento in questo elaborato di tesi è quella del NIST[1] nel documento *SP-800-145*[1] nel quale il cloud è presentato come un insieme di tecnologie aventi come obiettivo l'erogazione di servizi e risorse in modalità *on-demand* da un pool condiviso. La condizione di *outsourcing*, quindi, acquisisce una connotazione non strettamente necessaria all'adozione del servizio *cloud*, con cui tuttavia condivide alcuni vantaggi[2] soprattutto per realtà dove il *core business* non è il settore IT:

1. Contenimento dei costi
2. Velocità nel ciclo di sviluppo
3. Garanzia di prestazioni e di qualità
4. Servizio distribuito geograficamente
5. Contratti di affitto strutturati e dimensionati

[1] National Institute of Standards and Technology, *http://www.nist.org/*

## 1.2 Sicurezza nel cloud computing

Lo scopo finale dell'utilizzo di tecnologie *cloud* consiste nella possibilità per un'organizzazione di usufruire di un modello scalabile, elastico, standard, misurabile e orchestrabile al fine di poter garantire continuità di servizio e prestazioni elevate, demandando la gestione dei processi sistemistici a piattaforme centralizzate e intelligenti. A tal proposito il NIST[1] identifica tre modelli di servizio:

- **IaaS**, *Infrastructure as-a-Service*, nel quale è l'asset erogato è l'infrastruttura informatica, in termini di potenza di calcolo mediante sistemi di virtualizzazione, risorse di rete e storage. Essendo il modello più di difficile gestione, è spesso amministrato tramite un orchestratore. Esempi di tecnologie open-source in questo settore sono *OpenStack*[2], *oVirt*[3], *Apache CloudStack*

- **PaaS**, *Platform as-a-Service*, tramite il quale si fornisce all'utente la possibilità di eseguire servizi personalizzati offrendo meccanismi di contenimento nell'esecuzione, scalabilità e multi-tenancy. L'utente ha un controllo parziale sull'esecuzione del servizio: solitamente egli può interagire in modo limitato con il kernel. Una delle tecnologie più utilizzate è quella dei *container*, un evoluzione del concetto di *jail* proprio dei sistemi operativi BSD, il cui obiettivo è quello di utilizzare meccanismi di segregazione delle risorse basati sulle funzionalità del kernel. I servizi vengono eseguiti in un ambiente isolato, ed hanno un filesystem e uno stack di rete simulato in software dedicati. Una delle implementazioni più note della tecnologia *container* è *Docker*[4] che, nato inizialmente come evoluzione di LXC, è ora basato su una libreria proprietaria e costituisce la base per molte piattaforme PaaS (es. Kubernetes[5] e OpenShift[6]).

- **SaaS**, *Software-as-a-Service*, che permette all'utente di usufruire delle funzionalità di un singolo applicativo, riducendo al minimo l'effort computazionale sulla macchina dell'utente stesso. Tipicamente in questa categoria ricadono le applicazioni web, alcune applicazioni mobile e alcuni software per PC. Alcuni esempi di *SaaS* noti sono *Office 365 Online*[7] e *Google Docs*[8].

[2]Software open-source per la realizzazione di infrastrutture cloud pubbliche e private, https://www.openstack.org/

[3]Software open-source alla base della piattaforma

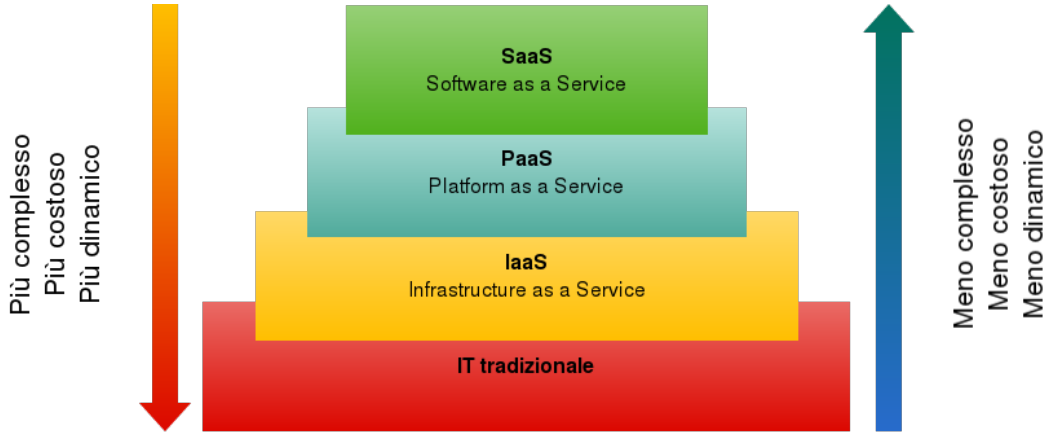
[4]Docker, *https://www.docker.com*

[5]Kubernetes, soluzione *PaaS* progettata da Google *https://kubernetes.io/*

[6]OpenShift, soluzione *PaaS* di Red Hat *https://www.openshift.com/*

[7]Office 365 è la versione cloud-based della nota suite per l'ufficio *Microsoft Office*, *https://www.office365.com*

[8]Google Docs è una suite per l'ufficio sviluppata da Google ed erogata esclusivamente come applicazione web, *https://docs.google.com*




:q!

Figura 1.1: Modelli di servizio

La parola chiave è quindi **"automazione"**. Questa, oltre a garantire una solidità del modello di distribuzione di un servizio grazie a schemi dichiarativi, apporta notevoli vantaggi anche dal punto di vista della sicurezza, facilitando la gestione degli aspetti di confidenzialità, integrità e disponibilità.
Il paradigma *as-a-service* ha infatti consentito la costituzione di una *baseline* robusta garantita dalla centralizzazione delle funzionalità di security le quali, essendo erogate come risorse *cloud*, sono interamente gestite dal *cloud service provider* - pubblico o privato - che può demandarne la gestione parziale all'utente mediante meccanismi di orchestrazione, interfacce grafiche ed API.
Se a primo impatto può apparire come un enorme vantaggio, di fatto ciò introduce un *single point of failure*, determinando livelli di rischio aggiuntivi rispetto alle infrastrutture tradizionali. Si pensi, ad esempio, alle funzionalità di *firewalling* offerte generalmente con la denominazione di *security groups* o Firewall as-a-Service (FWaaS): un'implementazione non idonea dal punto di vista funzionale nel substrato infrastrutturale del fornitore di servizi, potrebbe determinare la mancanza di sicurezza per i servizi che ne fanno affidamento. La stessa asserzione è valida per molte altre funzionalità comunemente offerte dal provider: cifratura dei volumi di storage, crittografia e controllo degli accessi nei servizi di block-storage e così via.

Ulteriori riflessioni possono essere fatte anche per quanto riguarda l'aspetto di integrità del dato: se da una parte il cloud service provider implementa già meccanismi di basso livello per la persistenza dello storage, ridondanza, sistemi di backup automatici, dall'altra non si ha la chiara evidenza di come questi aspetti siano effettivamente gestiti e di come la proprietà sia garantita.

Per quanto concerne la proprietà di disponibilità, la dicotomia va ricercata trattando i concetti di disponibilità del dato e disponibilità del servizio separatamente. Il *cloud computing* offre intrinsecamente solidità in quanto basato sui concetti di scalabilità, elasticità e ridondanza. Grazie ai meccanismi di orchestrazione tramite API è infatti possibile configurare le applicazioni per l'*auto-scaling*, al fine di mantenere una qualità adeguata nell'erogazione del servizio al crescere degli utenti. Ciò, dal punto di vista della sicurezza, ha portato a notevoli benefi-

ci per quanto riguarda la mitigazione di attacchi DoS[9], garantendo la continuità di servizio riducendo i costi. Tuttavia esistono dei prerequisiti per garantire la disponibilità: innanzitutto il *cloud service provider* deve assicurare la ridondanza dei dati e della rete, contemplando l'ipotesi di distribuire le risorse su più località geografiche, con l'obiettivo sia di prevenire guasti localizzati che di erogare la risorsa dalla località più vicina rispetto all'utente.

Nel momento in cui funzionalità comunemente demandate ad hardware specifico vengono implementano in software, si determinano sia benefici che svantaggi che devono sia essere contemplati in fase di valutazione del rischio che trattati nei contratti di service level agreement. Una compromissione dell'interfaccia di gestione della piattaforma cloud, sia che si tratti di una dashboard sia che si tratti di un'interfaccia API, può portare a un'interruzione di servizio.

Gli standard di sicurezza classici, così come l'assetto normativo e i contratti di *service level agreement*, necessitano di essere adeguati per supportare l'integrazione di tecnologie cloud all'interno degli stack tradizionali, tenendo conto delle problematiche di *shared responsability* presentate.

Il NIST [1] riconosce quattro diversi modelli di deployment:

- **Public Cloud**: modello in cui le risorse sono fornite per un utilizzo pubblico. È tipicamente erogato in outsourcing tramite la rete internet. L'hardware è in mano a un unico provider che eroga servizi in *outsourcing* e ne dispone le metriche e la tariffazione.

- **Private Cloud**: cloud dedicata a un'azienda o organizzazione, sfruttata per erogare servizi appartenenti al provider. L'hardware è generalmente nel datacenter dell'organizzazione.

- **Hybrid Cloud**: approccio ibrido dato dalla composizione di public cloud e private cloud, o di public cloud e infrastrutture tradizionali. Le infrastrutture coinvolte rimangono distinte e sono legate tra loro da un'unica tecnologia (standard o proprietaria) che facilita la migrazione e la portabilità delle risorse.

- **Community Cloud**: modello che fornisce una cloud per uso esclusivo di una comunità di utenti appartenenti ad organizzazioni con obiettivi funzionali comuni. Può essere di proprietà di una o più organizzazioni della community, o di terze parti.

Per ognuno di questi modelli è possibile esplicitare dei requisiti da soddisfare al fine di colmare il rapporto di sfiducia proprio di questo settore[3].

## 1.3 Valutazione del rischio: vulnerabilità, minacce e attacchi

In letteratura sono stati proposti molti lavori sulla valutazione del rischio su infrastrutture cloud. Nei paragrafi a seguire verranno discussi alcuni di questi approcci, sulla base della metodologia utilizzata da Ardagna et Al.[3]. Le vulnerabilità

[9] Denial of Service

possono essere categorizzate in tre macro aree, in base alla superficie di attacco considerata:

1. **Livello applicativo**: quando l'attacco è condotto da un qualsiasi attore nei confronti di una piattaforma SaaS
2. **Tenant su tenant**: quando l'attacco è condotto da attori appartenenti a un tenant nei confronti di un altro tenant
3. **Provider su tenant** e **Tenant su provider**: quando l'attacco è condotto dal provider nei confronti di un tenant (tipicamente malevolo) oppure da un tenant nei confronti del provider

### 1.3.1 Livello applicativo

Si tratta di vulnerabilità tradizionali che da anni tengono sotto scacco il panorama *web services*: si va da attacchi protocollari sulla comunicazione tra servizi fino alla compromissione di applicativi software specifici. Il target dell'attacco sono le piattaforme SaaS, spesso derivate dal porting di un'applicativo tradizionale sul cloud e non nativamente pensate per essere erogate online: per questo motivo sono caratterizzate da una superficie di attacco molto vasta.

Alcuni lavori significativi citati nel survey di rifermento [3] sono:

- **Gruschka and Iacono, 2009**[4], nel quale è stato presentato un *replay attack*, sfruttando una vulnerabilità del meccanismo di verifica della firma digitale sull'interfaccia SOAP di *Amazon EC2*, e sono state eseguiti comandi sulle API con i privilegi di un utente legittimo
- **Bugiel et Al., 2011**[5], che hanno analizzato le minacce sulla confidenzialità e la privacy estraendo con successo informazioni sensibili da immagini di macchine virtuali Amazon

### 1.3.2 Tenant su tenant

Le vulnerabilità *tenant su tenant* sono tipiche dei sistemi virtualizzati, quando tenant differenti condividono la stessa infrastruttura e, più specificatamente, lo stesso hardware fisico: gli attacchi possono avvenire per configurazioni erronee o vulnerabilità sull'infrastruttura di virtualizzazione. Si tratta quindi di attacchi che avvengono al livello più basso dello stack cloud[3].

Alcuni contributi interessanti per questa categoria di attacchi e vulnerabilità sono quelli di:

- **Ristenpart et al**, 2009[6], in cui è discusso un attacco alla confidenzialità delle informazioni relative a istanze di servizi in esecuzione. L'attacco dimostrato è basato sul fatto che i servizi sono ospitati sullo stesso hardware, per cui per un servizio è possibile generare traffico e monitorare le proprie performance per fare inferenza su quelle di un altro servizio.
- **Green**[7], che propone un ulteriore attacco di tipo *side-channel* che coinvolge, questa volta, due virtual machine ospitate sullo stesso hardware.

### 1.3.3 Provider su tenant, tenant su provider

Le vulnerabilità di questo tipo si verificano ogni qual volta un utente o un'organizzazione sposta le proprie risorse su un'infrastruttura cloud non fidata - nella quale il provider è malevolo oppure semplicemente curioso - oppure nel caso in cui l'utente inizia ad usare un servizio cloud con l'obiettivo di attaccare il provider (ad esempio creando botnet per lanciare attacchi denial of service, attaccando le API di orchestrazione e così via) [3].

Le tipologie di attacchi che sfruttano queste vulnerabilità, sono generalmente rivolte al livello IaaS[3], ma non è esclusa la possibilità di attacchi a livello PaaS e SaaS.

Il survey si sofferma sul lavoro Huan Liu[8], il quale illustra una attacco DDoS basato sulla saturazione della banda della rete virtuale: la virtualizzazione dello stack di rete a livello software (*software-defined network*) richiede, oltre a risorse di rete, anche un'elevata capacità di calcolo.

Le vulnerabilità provider su tenant sono invece trattate da Rocha e Correira[9] che propongono una panoramica dei possibili attacchi alla confidenzialità - che possono essere condotti anche dal fornitore di servizi - discutendone le contromisure, e da Bleikertz et al. [10] che si concentrano sulla problematica di proteggere i clienti da attacchi condotti da provider esterni, fornendo un'architettura *Cryptography as-a-Service client-driven*.

La problematica di confidenzialità nella casistica *provider-on-tenant* è anche l'oggetto di De Capitani di Vimercati et. al[11] in cui è descritta una tecnica per preservare la confidenzialità del dato riallocandolo in modo dinamico ad ogni accesso su tre nodi, risolvendo così anche i problemi di collusione tra i service provider coinvolti. Un ulteriore articolo di De Capitani di Vimercati et al[12]. affronta la problematica del provider *onesto ma curioso* con una soluzione per l'integrità dei risultati delle query di join, che discute la casistica di un server di storage di terze parti e di fornitori di potenza di calcolo esterni e malevoli i quali producono i risultati del join per basi di dati ospitate esternamente.

## 1.4 Tecniche di sicurezza per la cloud

Data l'etereogeneità delle problematiche e degli approcci adottati negli articoli citati, è possibile affermare che garantire proprietà di sicurezza in ambienti cloud è molto impegnativo: questi lavori presentano solamente soluzioni parziali al problema, affrontando di volta in volta problemi specifici e presentando tecniche sviluppate *ad-hoc*[3]. Saranno di seguito presentati alcuni approcci e tecniche per garantire la sicurezza su sistemi cloud.

### 1.4.1 Autenticazione e controllo degli accessi

I sistemi tradizionali per l'autenticazione e il controllo degli accessi si sono verificati inefficienti per la cloud, pertanto è stato necessario definire nuovi approcci. L'adozione di nuovi *pattern* di sviluppo orientati alla scalabilità - come ad esempio il pattern *micro-services*, naturale evoluzione delle architetture SOA - ha reso necessario sviluppare meccanismi di autenticazione decentralizzati e federati.

Almulia and Yeun [2010] offrono una panoramica sui protocolli di autenticazione e *identity management*, analizzandone la sicurezza, l'effort implementativo e i costi[13].

Costituendo parte critica per la maggior parte dei sistemi, i servizi di autenticazione, gestione dell'identità e gestione delle policy di accesso sono erogati *as-a-service*.

Takabi e Joshi hanno descritto un sistema di gestione delle policy *as-a-service* (PMaaS, *Policy Management as-a-service*) che fornisce un punto di controllo centralizzato indipendente dalla locazione della risorsa[14]. Prima di accedere a una risorsa è necessario contattare il server di autenticazione e autorizzazione centralizzato che rilascerà il *grant* dopo opportuna verifica. *Azure Active Directory*, il porting *SaaS* di Microsoft Active Directory, provvede sia a funzionalità di autenticazione che di policy management e fornisce alcuni driver di integrazione per la maggior parte dei protocolli noti.

Tuttavia, poiché molte realtà complesse dispongono già di meccanismi di autenticazione mediante *ticket granting* isolate dalla rete Internet, sono stati ideati anche modalità di autenticazione e controllo degli accessi completamente *stateless* (ad esempio OAuth). È il caso dei *JSON Web Token*, formalizzati nella RFC 7519: l'*authentication server*, dopo aver validato la richiesta di autenticazione, restituisce un token JSON firmato che contiene l'identità dell'utente e tutti i *grant* per le autorizzazioni ad esso relative. Non esiste il concetto di sessione, la validità del token è data esclusivamente da una marca temporale e da una durata. Il token può essere utilizzato quindi per autenticare le richieste verso i vari servizi, cui spetta l'onere di verificarne la validità del contenuto e della firma, decifrabile tramite segreto condiviso con il server di autenticazione che lo ha emesso. I vantaggi di un approccio simile sono molteplici, tuttavia è impossibile revocare il token una volta emesso. Eventuali blocchi sono effettuabili tramite sistemi di *blacklisting* che riporterebbero in auge la problematica della decentralizzazione che si voleva risolvere. La prassi è quindi quella di emettere token one-time o con durata breve, al fine di minimizzare la durata di una possibile finestra temporale di attacco.

### 1.4.2 Crittografia, firma digitale e trusted computing

La crittografia è essenzialmente utilizzata per proteggere la confidenzialità dei dati, delle comunicazioni e le attività sensibili da tutti quegli avversari che mirano a disturbare l'operatività della cloud. La maggior parte della letteratura utilizza tecniche di crittografia per preservare la confidenzialità: l'obiettivo di queste metodologie è di facilitare la migrazione dei dati gestiti da sistemi tradizionali verso la cloud. Tuttavia non sono assenti tecniche focalizzate su altre proprietà di sicurezza, come l'utilizzo della firma digitale per curare gli aspetti di integrità e privacy.

#### Trusted Computing

Il *trusting computing* è una tecnica utilizzata per effettuare computazioni sicure, basata sull'utilizzo della crittografia asimmetrica e di un dispositivo hardware

dedicato (TPM, Trusted Platform Module) tramite il quale è possibile *i)* identificare univocamente i dispositivi con un numero di serie e una chiave di cifratura implementata in hardware *ii)* cifrare informazioni con la chiave di cifratura *iii)* firmare informazioni con la chiave di cifratura. Queste funzionalità pongono le basi per una serie di utilizzi avanzati volti a preservare l'integrità e la confidenzialità di dati - sia in transito su una rete, che memorizzati su disco o sui firmware del dispositivo - codice e hardware, riducendo o annichilendo gli effetti di eventuali attacchi.

Boampong e Wahsheh nel 2012 hanno proposto un modello per utilizzare il TPM al fine di garantire la correttezza dei processi di autenticazione, l'integrità e la confidenzialità sulla cloud[15]. Portare il TPM sul cloud significa realizzarne una versione virtuale, così come illustrato da Krautheim[16] nel 2009, basandosi sul concetto di virtual-TPM (vTPM) già descritto da Berger et al. nel 2006[17]. Il vTPM è un componente software che implementa le stesse funzionalità del TPM hardware, garantendo la multi-tenancy mediante istanze multiple e multiplexing. I vantaggi dell'utilizzo di una tecnologia di *trusted computing* nel contesto cloud sono molteplici, come la possibilità per l'utente di fare enforcement di politiche di privacy togliendo la possibilità al cloud service provider di modificarle, fornendo una soluzione parziale problematiche di *shared responsability* discusse. Come illustrato da Velten and Stumpf[18] e più recentemente da Szefer e Lee[19] il TPM può essere utilizzato per garantire confidenzialità e integrità a tutti i livelli dello stack, prevenendo tampering da parte del fornitore di servizi e attacchi da parte di altri tenant o da malware.

## 1.5 Approcci per assurance, testing, monitoraggio e compliance

I progressi nella ricerca sulla sicurezza della cloud hanno portato la necessità di avere tecniche di *security assurance* per aumentare la confidenza degli utenti nei confronti del provider[20]. Per *assurance* si intende la modalità per ottenere, con un certo livello di precisione, la consapevolezza che l'infrastruttura e/o le applicazioni manterranno nel tempo una o più proprietà di sicurezza, e la loro operatività non sarà compromessa indipendentemente da malfunzionamenti o attacchi[21]. In accordo con Ardagna et Al.[3], è possibile affermare che quello di *assurance* è un concetto più esteso della mera nozione di *sicurezza informatica*, comunemente definita come *la protezione delle informazioni e dei sistemi informativi da accessi, utilizzi disclosure, interruzioni del funzionamento, modifiche e distruzioni non autorizzate*. Nella cloud è molto facile avere livelli di sicurezza elevati con livelli di assurance scarsi poiché le funzionalità di sicurezza realmente implementate sono difficilmente percepite.

Per la messa sicurezza delle realtà che decidono di trasferire degli *asset* sulla cloud è necessario considerare tre aspetti fondamentali:

- Necessità di una soluzione di analisi e gestione del rischio, in grado di valutare l'impatto dell'adozione di servizi cloud sul business

- Esigenze di *transparency*, ovvero la possibilità per l'utente di essere consapevole del modello di business del fornitore di servizi
- Soluzione di assessment, verifica delle policy e della compliance, che permetta sia di verificare lo stato istantaneo del livello di conformità, che di spiegare all'utente le metodologie attuate per mantenere livelli di compliance adeguati, secondo il principio "comply-or-exmplain" di MacNeil and Li[22]

L'obiettivo di questo lavoro di tesi è quello di fornire un framework per la security assurance i) insistendo sulla valutazione continuativa dello stato di sicurezza sulla cloud ii) offrendo un framework cloud-based per la security assurance insistendo su

- testing di proprietà non funzionali
- monitoraggio continuativo della sicurezza del sistema
- conformità del sistema a politiche di sicurezza, siano esse definite internamente ad un'organizzazione, siano esse provenienti da uno standard di settore
- ottemperare alle esigenze di transparency degli utenti della cloud, offrendo una dashboard panoramica sullo stato della cloud del provider

### 1.5.1 Testing di proprietà non funzionali

Il *testing* è definito come la fase del ciclo di vita del software composta da tutte le attività, statiche o dinamiche, atte a determinare che questo soddisfi i requisiti specificati e che sia conforme all'obiettivo proposto, nonché per rilevare eventuali difetti.

Nel contesto *cloud* possiamo riconoscere due tipologie di soluzioni di testing: quelle specifiche per il collaudo di infrastrutture cloud e quelle generiche per il testing del software, applicabili anche a servizi cloud.

Il lavoro di tesi si focalizzerà maggiormente sulla prima categoria insistendo sulla validazione delle proprietà a tutti i livelli dello stack (in accordo con Riungu et. Al[23]); nonostante ciò il framework proposto può essere adattato ad entrambe le tipologie.

### 1.5.2 Monitoraggio continuativo della sicurezza del sistema

La natura stessa dei sistemi cloud complica notevolmente l'analisi delle informazioni relative allo stato dei servizi: a causa dell'elevata complessità dei software impiegati nell'orchestrazione e nell'erogazione delle risorse è spesso difficile rilevare cambiamenti nello stato del sistema, il cui back-end è continuamente tempestato di eventi. È quindi necessario introdurre una componente di monitoraggio, collezionamento e correlazione di eventi.

Per valutare aspetti non funzionali come la sicurezza, è poi necessario che questi eventi vengano contestualizzati: possono essere necessarie pertanto analitiche *stateful*, effettuabili anche tramite strumenti più complessi o provenienti dal mondo *big-data*.

Proprio per facilitare scenari di *software integration* il framework proposto nei prossimi capitoli è stato strutturato esasperando la modularità, ed è stato basato principalmente su tecnologie *open-source*.

Come per il testing, anche per il monitoraggio è possibile individuare sia soluzioni generiche sia soluzioni specifiche per il mondo *cloud*. Software come *Nagios*[10] e *Ganglia*[11] rientrano nella prima categoria, ma vantano livelli di espandibilità tali da poter essere adeguati ai sistemi di collezionamento delle metriche dei maggiori software cloud. Ulteriori soluzioni come *Sensu*[12], *Sysdig*[13], *Weave*[14] contengono strumenti specifici per la cloud.

Per quanto riguarda gli aspetti di sicurezza, la disponibilità di potenza computazionale on-demand, ha garantito la possibilità di effettuare il deploy scalabile di sistemi IDS[15] e IPS[16]. L'utilizzo di questa tipologia di software è stato approfondito da Modi et al. [24], i quali hanno illustrato come utilizzarli sulla *cloud* al fine di mitigare le diverse tipologie di attacchi al paradigma CIA (attacchi provenienti dall'interno, dall'esterno, attacchi di flooding, *privilege escalation*, *port scanning*, attacchi agli *hypervisor* di virtualizzazione e attacchi tramite *backdoor*). Un lavoro di Ficco et Al. del 2013[25] ha presentato un'architettura multi-layer per il rilevamento delle intrusioni, che supporta l'aggregazione di eventi complessi.

Lavori successivi hanno successivamente presentato approcci più specifici e focalizzati su problemi singoli, come Ardagna et Al. 2014[20] che tramite un approccio introspettivo sulle virtual-machine ha prodotto un meccanismo di rilevazione dei *rootkit*.

### 1.5.3 Conformità del sistema a politiche di sicurezza e cloud transparency

Il presente lavoro di tesi trova le sue origini nel progetto europeo FP7 CUMULUS[26] (Certification infrastrUcture for MUlti-layer cloUd Services), nel quale sono stati proposti modelli, processi e strumenti a supporto di un processo di certificazione per proprietà di sicurezza e non-funzionali in ambito di cloud computing. L'obiettivo del processo di certificazione è quello di fornire quante più evidenze possibili per attestare che un sistema software garantisca determinate proprietà non funzionali e si comporti in modo corretto[3]; si tratta di un approccio alla sicurezza già sperimentato in altri ambiti che tuttavia rimane di difficile applicazione nel contesto dei *web-services*, in particolare nella cloud[27]. Infatti, le tecniche di certificazione usuali che considerano il software come blocco monolitico, vanno a scontrarsi con una struttura complessa e *multi-tier*[27] e necessitano

[10]Nagios, piattaforma di monitoraggio distribuita general purpose, *http://www.nagios.org/*

[11]Ganglia, soluzione per il monitoraggio delle performance dei cluster in ambito grid computing, *http://ganglia.sourceforge.net*

[12]Sensu, *https://www.sensuapp.org/*

[13]Sysdig, sistema per l'identificazione dei problemi nei sistemi basati su containerhttp://www.sysdig.org

[14]Weave, piattaforma SaaS per il monitoraggio di architetture a micro-servizi *https://www.weave.works/*

[15]*Intrusion Detection System*, software per il rilevamento delle intrusioni

[16]*Intrusion Prevention System*, sistemi preventivi per la rilevazione di attività anomale usati per prevenire incidenti informatici

di essere integrate con i processi e caratteristiche tipiche del mondo cloud, come il deployment, la discovery degli asset, l'elasticità e il paradgima on-demand. Il problema è stato dapprima affrontato in Damiani et al. [2009b] [28], in cui è definita una soluzione di certificazione per i servizi basata su certificati di sicurezza basati su test-case firmati. Più recentemente Anisetti et. al [29][30][31] hanno proposto uno schema di certificazione sulla base di un processo di testing basato su un modello, esteso poi con un processo di certificazione incrementale al fine di coprire le esigenze evolutive del paradigma dei servizi.

L'obiettivo di questa tesi, tuttavia, non è quello di fornire un meccanismo di certificazione, bensì quello di offrire un framework per il controllo della conformità di un sistema rispetto alle proprietà non funzionali attese. La metodologia utilizzata è basata sul concetto di *auditing*, ovvero la possibilità di verificare il comportamento di un sistema per valutarne l'adeguatezza rispetto alle policy dell'utente piuttosto che ai regolamenti o alle legi vigenti.[3] Si vuole quindi rendere la cloud *auditable* - al fine di ottemperare alle esigenze di transparency dell'utente, incrementando così il livello di fiducia dell'utente nei confronti del provider e permettendo allo stesso di essere in grado di effettuare scelte ponderate delle varie soluzioni rispetto ai propri requisiti, funzionali e non.

La *transparency* consiste, per l'appunto, nel concedere all'utente una visione di alto livello a dati aggregati ed evidenze collezionati dal provider stesso a basso livello, ed è considerato alla base di ogni approccio efficace per la cloud assurance[20][32].

L'assenza di *transparency* infatti rende i problemi di sicurezza difficilmente percettibili per l'utente[3], in quanto i contratti di *service level agreement* non forniscono parametri tecnici per misurare il livello di sicurezza delle applicazioni e dei dati ospitati sulla cloud[33].

Essa, inoltre, è fondamentale per supportare sia una visione dei processi interni da parte del provider che una visione dei processi esterni per il cliente per fini di sicurezza, così da bilanciare entrambe le esigenze[3].

## 1.6 Conclusioni

Finora la *security assurance* è effettuata mediante tecniche perlopiù manuali e dall'effort elevato, con cadenze trimestrali o semestrali: il processo di verifica non è effettuato con continuità.

Nel prossimo capitolo verrà presentato Moon Cloud, un framework automatico e programmabile per documentare, valutare, osservare dei controlli tecnici (auditing su controllo degli accessi, configurazione del sistema, crittografia ecc.), controlli di processo (analisi delle vulnerabilità, analisi del rischio, acquisizione di evidenze sul funzionamento di sistemi e servizi) e controlli di sistema (gestione delle configurazioni, consapevolezza e training, gestione delle modifiche e dei cambiamenti)

Successivamente verrà illustrato ed analizzato FedRAMP, il programma governativo americano che fornisce un approccio standard per effettuare il *security assessment* e automatizzare il monitoraggio continuativo dei servizi cloud; verrà mostrata l'integrazione dello stesso nella soluzione Moon Cloud, illustrando gli

approcci per l'inclusione dei controlli di sicurezza di FedRAMP, per concludere con la valutazione e la validazione del lavoro mediante l'assessment del deployment di un'architettura software a microservizi (Moon Cloud stesso) in modalità multi-layer.

# Capitolo 2

# Moon Cloud: un framework per il monitoraggio e la security assurance

## 2.1 Introduzione

In questo capitolo verrà approfondito Moon Cloud[1], un framework per il monitoraggio e l'assurance di sistemi tradizionali, cloud e Internet of Things, sviluppato dal laboratorio SESAR Lab[2] dell'Università degli Studi di Milano.

L'obiettivo del progetto Moon Cloud è quello di implementare una metodologia automatica per la valutazione della sicurezza, delle performance e di altre proprietà non funzionali, offrendo un processo di assurance basato su attività di auditing e raccolta di evidenze.

Le caratteristiche di Moon Cloud sono le seguenti[34]:

- **Framework automatico e personalizzabile orientato a microservizi** basato su modelli per la raccolta di evidenze
- **Copertura di tutto lo stack cloud**, ai livelli *IaaS*, *PaaS*, *SaaS*
- **Possibilità di integrazione con tecnologie pre-esistenti e di terze parti**

L'orchestrazione avviene tramite una *dashboard* grafica erogata in modalità *Software as-a-Service*, la quale si interfaccia con diversi componenti che ne implementano la logica di funzionamento, l'esecuzione dei test, il collezionamento dei risultati e il reporting dello stato di compliance del sistema analizzato.

Moon Cloud nasce sulla scia del progetto europeo FP7 CUMULUS - il cui obiettivo è quello di fornire un framework per la certificazione di servizi cloud mediante l'assessment di proprietà non funzionali[26] - da cui riprende alcune caratteristiche architetturali.

L'obiettivo di questo capitolo è quello di dare al lettore il know-how necessario a comprendere le motivazioni di alcune decisioni prese nella realizzazione della tesi, in particolar modo per gestire alcune limitazioni del framework Moon Cloud. Di seguito sarà proposta una breve panoramica sulla terminologia utilizzata, sulle componenti principali, e sui principi di funzionamento dei processi di *assessment* e *compliance* all'interno della piattaforma.

[1] MOnitoring and assurance ON Cloud, *https://www.moon-cloud.eu/*

[2] SEcure Service-oriented Architectures Research Lab - *http://sesar.di.unimi.it*

## 2.2 Terminologia

- **Metriche**: insieme di proprietà non funzionali di cui si vogliono ottenere **misurazioni**.
- **Controllo**: rappresenta la modalità di raccolta ed elaborazione delle *misurazioni* di una *metrica*. Esso è descritto tramite un documento *JSON*[3], i cui attributi sono
  - **name**, nome del controllo
  - **description**, descrizione del controllo
  - **category**, categoria di appartenenza
  - **driver-name**, nome del driver che ne implementa il flusso di esecuzione
  - **inputs**, dati ricevuti in input
  - **outputs**, dati attesi in output (*misurazioni*)
- **Driver**: la porzione di codice che implementa il controllo
- **Test**: istanza di un controllo, che ne rappresenta l'esecuzione con gli input effettivi
- **Regola di valutazione astratta**, o *AER (abstract evaluation rule)*, regola logica che implementa un processo di valutazione. È data dall'aggregazione di controlli mediante operatori *booleani*. I suoi termini possono essere *controlli* e altre *AER*.
- **Regola di valutazione concreta**, o *ER (evaluation rule)*: istanza di una *AER*. I suoi termini possono essere *test* (mappati sui rispettivi *controlli* o altre *ER*.

[3]JSON, JavaScript Object Notation, *http://www.json.org/*

## 2.3 Architettura e componenti

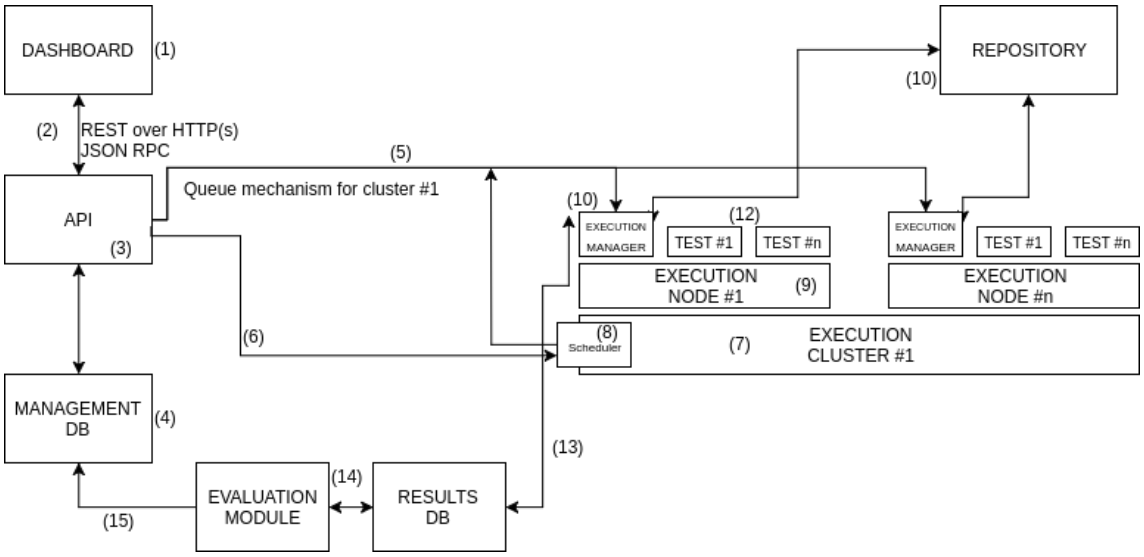



Figura 2.1: Architettura e componenti di Moon Cloud

In figura 2.1 sono illustrate l'architettura del framework Moon Cloud e l'interazione tra i vari microservizi che lo compongono. Questa può essere divisa in due aree. La prima, accessibile dagli utenti del sistema fornisce le funzionalità di gestione e comprende:

- **Dashboard** (1), sviluppata in Javascript, HTML5 e CSS mediante il framework Angular.JS. Rappresenta il punto di ingresso per l'utente, e fornisce un'interfaccia grafica per la fruizione delle funzionalità del prodotto. La *Dashboard* è supportata da API HTTP servite in modo sicuro tramite il protocollo TLS (2).
- **API** (3), costituiscono l'intefaccia con le principali funzionalità del sistema. Interagiscono con un database di management (4) seguendo il paradigma RESTful, e orchestrano l'esecuzione dei test. Il test può essere lanciato:
  - in modalità one-shot, inviando un messaggio agli *execution manager* (9) mediante un meccanismo di comunicazione basato su code(5).
  - inserendo un *task* periodico(6) nello *scheduler* (7)

  Successivamente alla conclusione del test di sicurezza, *one-shot* o *periodico* che sia, i risultati restituiti saranno disponibili sulla dashboard, con la possibilità di filtrarli e analizzarne le metriche.

La seconda invece, rappresenta il vero e proprio backend del framework, gestendo i meccanismi di esecuzione dei test, raccolta dei risultati e valutazione degli stessi. È composta da:

- **Execution Cluster** (8), ovvero l'insieme dei nodi - *execution node*, (9) - che eseguono materialmente il test. I cluster sono equipaggiati da uno *scheduler*

(7), che gestisce l'esecuzione temporizzata dei test periodici, per effettuare il monitoraggio in modo continuo, inviando i test in esso memorizzati a cadenze temporali scandite da un pattern *CRON*[4].

- **Execution Manager** (10), ovvero il servizio che gestisce il lavoro di ciascun nodo del cluster. Gli *execution manager* di uno stesso cluster sono connessi alla stessa coda: all'arrivo di un messaggio essi ne effettuano il *prefetch*, ne validano il contenuto, effettuano il download il driver relativo al controllo referenziato dal test dal repository (11), eseguono lo stesso con gli input contenuti nel messaggio, e collezionano gli output del test in un database non relazionale *time series*(12).

- **Evaluation Module** (13), rimane in ascolto sul database *time-series* (12) in attesa di eventi. Quando lo stato di uno specifico test cambia, avvia la procedura di valutazione di tutte le *ER* che fanno riferimento a tale test, e ne memorizza il risultato nel database di management (4).

- **Repository** (10), contiene i driver per i controlli di sicurezza. È realizzato tramite un ambiente Git[5], di cui condivide le funzionalità, e da un *registry* di *container*, che rappresentano il singolo driver. Il processo di sviluppo di un driver è ultimato da un motore di *continuous integration* che contestualmente al caricamento del codice dello stesso sul *repository* effettua la build del *container* associato e test automatici di validazione e sanitizzazione dello stesso, per poi pubblicare il *container* sul *registry*..

## 2.4 Moon Cloud come strumento di verifica delle *recommendation*

Mediante i componenti software illustrati nel paragrafo precedente, Moon Cloud è in grado di effettuare un processo di *security assessment* basato sulla verifica di *recommendation*. Per verifica delle *recommendation* si intende il controllo della conformità di un sistema target rispetto ad una data raccomandazione.

Si tratta di un processo complesso, che richiede l'aggregazione delle valutazioni di diversi servizi. La tecnica adottata da Moon Cloud consiste nella raccolta di evidenze mediante il testing e il monitoraggio di uno specifico servizio, al fine di verificare se la raccomandazione sia effettivamente rispettata oppure no.

[**Recommendation verification**] Sia la *recommendation* da verificare, la *recommendation verification* è una funzione$\hat{}$definita sulla tupla $\langle , \rangle$ che restituisce un valore booleano {true, false} dove:

- è l'insieme di *processi di valutazione* {} da eseguire. L'*output* del processo di valutazione è un valore *booleano* che espreime se la valutazione abbia avuto successo o no, aggregato alle *evidence* collezionate a supporto del risultato.

[4]http://man7.org/linux/man-pages/man5/crontab.5.html
[5]Git, *https://git-scm.com/*

- è una *evaluation rule* espressa come formula di logica proposizionale, i cui termini sono i singoli processi di valutazione . Essa combina il risultato di vari processi $\in$ e restituisce true se la valutazione ha successo, altrimenti restituisce false.

**[Eval{}]** Un processo di valutazione $Eval\{\}$ è una tupla della forma $\langle t, C \rangle$, dove:

- $t$ è il *ToE (Target of Evaluation)*, ovvero i servizi o i meccanismi, che costituiscono il perimetro entro cui la proprietà o la funzionalità di sicurezza deve essere valutata

- $C$ è il *Control*, ovvero la funzione di valutazione su $t$ che restituisce il risultato della valutazione insieme a un insieme di evidenze.

Un controllo $C$ specifica i dettagli su come collezionare le evidenze su un target $t$ per valutare la *reccomendation*. È definito nel seguente modo:
[$C$] $C$ è definito su una tripla della forma $\langle \phi, \lambda, \pi \rangle$, dove:

- $\phi$ è il flusso di esecuzione del processo di raccolta delle evidenze. È composto da una sequenza di operazioni atomiche.

- $\lambda$ è un insieme di *parametri* necessari a collegare il flusso $\phi$ al target $t$

- $\pi$ è un insieme di *Environmental Settings* che descrivono le caratteristiche dell'ambiente in cui il controllo deve essere eseguito e le possibili dipendenze software dello stesso.

Moon Cloud è in grado di eseguire molteplici processi di valutazione in parallelo, ciascuno dei quali si riferisce a un insieme di raccomandazioni che devono essere valutati. I controlli $C$ sono modellati utilizzando degli schemi, il flusso di esecuzione del codice ($\phi$) è modellato come una catena formata da tutte le operazioni che un controllo necessita di effettuare per raccogliere le evidenze, ed è implementato sotto forma di script Python. I parametri ($\lambda$) e l'ambiente ($\pi$) sono rappresentati da metadati; ciascuna operazione del flusso $\phi$ è collegata agli specifici parametri necessari per la valutazione, mentre l'ambiente $\pi$ rappresenta l'insieme di prerequisiti e dipendenze che devono persistere per l'esecuzione del controllo.

### 2.4.1 Regole di valutazione

Nel contesto implementativo le regole di valutazione sono modellate attraverso le classi *AbstractEvaluationRule* e *EvaluationRule*, rispettivamente per le regole astratte e per le regole concrete. Di seguito verranno approfondite entrambe le classi.

#### Regole di valutazione astratte

Per la classe *AbstractEvaluationRule* è definita una proprietà *formula* che contiene la formula in logica proposizionale che rappresenta il processo di valutazione.

La formula è espressa in un linguaggio libero dal contesto la cui grammatica è definita tramite la seguente BNF[6]:

```
expressions
    : expr $
    ;

expr
    : TOKEN_VAR
    | expr TOKEN_AND expr
    | expr TOKEN_OR expr
    | expr TOKEN_IMPLIES expr
    | expr TOKEN_IFF expr
    | TOKEN_NOT expr
    | TOKEN_LPAREN expr TOKEN_RPAREN
    ;
```

I token di questa grammatica sono *variabili* gestite dall'espressione regolare

- TOKEN_VAR → *[cf]"#"\d+*

e *operatori* (distinguibili in operatori unari e binari a seconda dell'arietà), implementati nel seguente modo:

- TOKEN_NOT → not(expr), operatore unario che effettua la negazione del termine: $return \sim expr$
- TOKEN_AND → and(expr1, expr2), operatore binario che effettua l'*and* logico dei termini: $return\ expr1 \wedge expr2$
- TOKEN_OR → or(expr1, expr2), operatore binario che effettua l'*or* logico dei termini: $return\ expr1 \vee expr2$
- TOKEN_IMPLIES → implies(term1, term2), operatore binario che implementa l'operatore di implicazione: $return \sim expr1 \vee expr2$
- TOKEN_IFF → iff(term1, term2), operatore binario che implementa l'operatore "se e solo se": $return\ (\sim term1 \vee term2) \wedge (\sim term2 \vee term1)$

*TOKEN_LPAREN* e *TOKEN_RPAREN* sono rispettivamente la parentesi tonda aperta e chiusa.

Tramite questo linguaggio è possibile generare tutte le formule ben formate della logica proposizionale.

Le variabili possono essere quindi *Controlli* (rappresentati dalla classe *Control*, e indicati nella formula con la sintassi "c#<id>", dove <id> è l'identificativo del controllo) ed altre *AER* (indicate nella formula, in modo analogo ai controlli, con la sintassi "f#<id>"). Ciò permette di organizzare le regole di valutazione astratte secondo una gerarchia, concedendo all'utente una maggiore capacità espressiva nel processo di traduzione da politica a regola logica.

[6]BNF, Backus-Naur Form

**Regole di valutazione concrete**

Le regole di valutazione concrete rappresentano le istanze delle *AER* sopra descritte. In questo caso la proprietà *formula* è derivata mediante l'applicazione la funzione di mapping $m(aer)$, (2.1) dove $aer$ è la regola di valutazione astratta di partenza, $c_i$ è un controllo, $t_i$ è un test e $e$ è una regola di valutazione concreta.

$$m(aer) : c \rightarrow t \; \forall c_i \in aer \mid t \text{ refers } c \text{ , } f \rightarrow e \; \forall aer_i \in aer \mid e \text{ refers } aer_i \qquad (2.1)$$

Tutte le regole di valutazione astratte incluse nella regola di partenza sono quindi sostituite da una o più regole di valutazione concrete che le referenziano, analogamente tutti i controlli inclusi nella regola di partenza sono sostituiti da uno o più test. Il mapping avviene sulla base di un oggetto JSON *chiave-valore*, in cui la chiave rappresenta l'identificativo dell'elemento di partenza e il valore rappresenta l'identificativo dell'elemento di arrivo.

La grammatica utilizzata per la validazione della formula così derivata, è ovviamente analoga alla precedente, ad eccezione del token *TOKEN_VAR* che assume valori nella seguente espressione regolare:

$$\text{TOKEN\_VAR} \rightarrow \textit{[et]"\#"\textbackslash d+}$$

supportando il carattere '*e*' per indicare le Evaluation Rule, e il carattere '*t*' per indicare i test. In fase di esecuzione del processo di valutazione sarà questa la formula effettivamente utilizzata, sostituendo a ciascun termine il valore booleano del test o dell'evaluation rule referenziata.

### 2.4.2 Driver per i controlli di sicurezza

Di seguito verrà illustrata la struttura di un driver Moon Cloud, utilizzato per l'implementazione effettiva dei controlli di sicurezza e per l'esecuzione dei test.

I driver sono costituiti da container *Docker* basati sull'immagine *python:2-onbuild*[7] e possono essere eseguiti sia all'interno del framework Moon Cloud che in modalità completamente standalone. Il container è gestito da un *entrypoint* che riceve gli input nello stream *STDIN*[8], colleziona eventuali log di diagnostica in *STDERR*[9] e restituisce gli output in *STDOUT*[10], secondo lo standard POSIX.

Sia l'input che l'output di un driver sono costituiti da documenti JSON le cui chiavi devono corrispondere a quelle dichiarate negli attributi *inputs* e *outputs*, del *Controllo* corrispondente, precedentemente trattati nella sezione 2.2

#### Struttura di un driver

All'interno del container sono presenti due componenti:

- *entrypoint.py*, il cui ruolo è quello di leggere ed effettuare la deserializzazione degli *input* da *STDIN*, leggere i metadati del driver, e precaricare la classe del driver. Il discovery della classe viene fatto in modo automatico da una directory chiamata *test/* che andremo a dettagliare tra poco.

- *driver.py*, che fornisce un'*interfaccia* mediante la classe *Driver* che ciascun payload dovrà implementare. Una peculiarità della classe *Driver* è la presenza della sottoclasse *__metaclass__* che fornisce un meccanismo di *subscription* per tutte le classi che ereditano da Driver. Questo permette di far funzionare il meccanismo di *autodiscovery*, permettendo a chi scrive il payload di lavorare in modo agnostico rispetto alle funzionalità del framework. L'esecuzione effettiva del codice avviene mediante il metodo *run* e un meccanismo di gestione delle operazioni atomiche in due flussi (forward e rollback) dettagliatamente illustrati nella sezione successiva. Un'altra classe molto importante è la classe *DriverResult* che fornisce la possibilità di restituire le misurazioni effettuate dal controllo in forma *chiave-valore*.

Altri due file molto importanti sono:

- *Dockerfile*, che costituisce la definizione di ogni container Docker. Questo permette di installare o compilare eventuali dipendenze software utili all'esecuzione del driver

- *requirements.txt*, che, in Python, contiene tutte le librerie del database PyPI-https://pypi.python.org/ da cui il codice del driver dipende.

[7] https://hub.docker.com/_/python/
[8] Standard Input
[9] Standard Error
[10] Standard Output

**Payload del driver**

Il payload del driver è contenuto nella directory *test*, ed è costituito da una classe che eredita la classe *Driver* illustrata in 2.4.2 e ne implementa il metodo di interfaccia *appendAtomics*.

Esso è definito da:

- una sequenza $A$ di $n$ operazioni atomiche $o_n$, con corrispondenza biunivoca con le chiavi specificate nel documento JSON di input

$$A = \{o_0, o_1, o_2, ..., o_n\}$$

- una sequenza di $n$ operazioni di rollback - una per ogni operazione atomica - che svolgono azioni di *undo*

$$R = \neg A = \{r_0, r_1, r_2, ..., r_n\}$$

aggregati in coppie ordinate.

$$P = \{(o_0, r_0), (o_1, r_1), (o_2, r_2), ..., (o_n, r_n)\}$$

Ogni operazione atomica $o_x \in A$ riceve in input gli output dell'operazione $o_{x-1}$; analogamente ogni operazione $r_x$ di rollback $r \in R$ riceve in input gli output di $r_{x+1}$.

$$R_1 \subseteq R = \{r_{x-1}, r_(x-2), ..., r_{x-n}\}$$

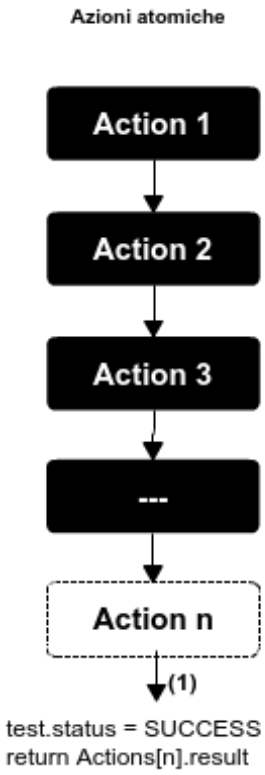



Figura 2.2: Esecuzione corretta di un test

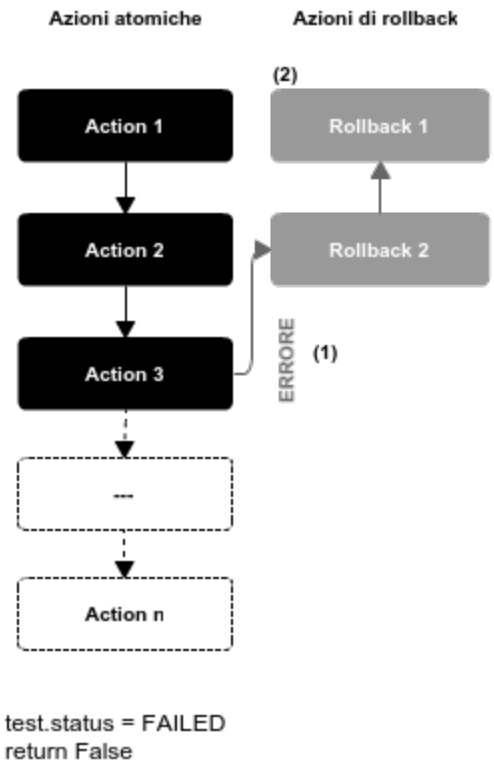



Figura 2.3: Esecuzione di un test con operazioni di rollback

Nella figura 2.2 è mostrata l'esecuzione di un driver di sonda il cui test viene eseguito correttamente. Il risultato dell'ultima operazione atomica deve essere un valore *boolean* e viene utilizzato come risultato finale (1).

In caso di fallimento di un'operazione $o_x \in P$ , $x \leq n$ (figura 2.3) (1) l'interfaccia della classe *Driver* si occupa di lasciare il sistema target in uno stato sicuro provvedendo ad eseguire in sequenza le operazioni di rollback e restituendo come risultato *False*.

Gli input globali del test sono disponibili interrogando la proprietà "*testinstances*", un dizionario a due livelli che contiene nella chiave di primo livello il nome della fase a cui l'input è riferito, nella chiave di secondo livello il nome della chiave del valore di input.

```
1
2  from driver import Driver
3  from time import sleep
4
5  class MyDriver(Driver):
6      def step1(self, inputs):
7          self.logger.info("Sto eseguendo lo step 1")
8          sleep(10)
9          self.logger.info("Ho eseguito lo step 1")
10         return True
11
12     def rollback1(self, inputs):
13         self.logger.info("Sto eseguendo il rollback dello step 1")
14         sleep(10)
15         self.logger.info("Ho eseguito il rollback per lo step 1")
16
17     def step2(self, inputs):
18         self.logger.info("Sto eseguendo lo step 2")
19         self.logger.info("Stampo gli input del test")
20         self.logger.info(json.dumps(self.testinstances))
21         sleep(10)
22         self.logger.info("Ho eseguito lo step 2")
23
24     def rollback2(self, inputs):
25         self.logger.info("Sto eseguendo il rollback per lo step 2")
26         sleep(10)
27         self.logger.info("Ho eseguito il rollback per lo step 2")
28
29     def appendAtomics(self):
30         self.appendAtomic(self.step1, self.rollback1)
31         self.appendAtomic(self.step2, self.rollback2)
```

Nella directory "test", oltre al payload del driver, è contenuto un file *metadata.json* con le seguenti chiavi:

- driver-name, nome del driver
- inputs, oggetto di cui le chiavi costituiscono il nome simbolico da associare all'input e i valori costituiscono il tipo di dato atteso (stringa, intero ed eventuali tipi di dato personalizzati, anche complessi, come indirizzo ip, account Amazon, account OpenStack)
- outputs, oggetto analogo ad inputs, ma per la gestione degli output
- schema, oggetto contenente lo JSON schema utilizzato per effettuare il rendering del form di input nella dashboard

## 2.5 Esempio

Verrà di seguito proposto un esempio per illustrare la definizione completa di un processo di valutazione all'interno della piattaforma Moon Cloud. La proprietà analizzata da questo esempio è la *confidenzialità del dato* ottenuta dall'aggregazione in and logico di due diversi controlli:

1. Confidenzialità dello storage (*c#1*)
2. Confidenzialità del canale di comunicazione (*c#2*)

### 2.5.1 Abstract Evaluation Rule

```
c#1 and c#2
```

```
{
    "id":1,
    "name":"Data Confidentiality",
    "description":"It allows to execute a set of tests againt a given service to ensure
        that information are provided over a secure channel and stored over a secure
        storage.",
    "category":[],
    "formula":"c#1 and c#2",
    "enforced_control":null,
    "enforced_operator":null,
    "cardinality":null,
    "related_controls":[1, 2],
    "related_aers":[],
    "metadata":null
}
```

### 2.5.2 Controllo

#### Confidenzialità dello storage

```
{
  "category" : [  ],
  "driver" : "storage-confidentiality",
  "id" : 1,
  "metadata" : {
      "description" : "Evaluates SSL configuration for a given target",
      "schema" : { "config" : {
              "properties" : {
                  "ssh_string" : {
                      "title" : "SSH Connection String",
                      "type" : "string"
                    },
                  "ssh_key" : {
                      "default" : 80,
                      "title" : "SSH Key",
                      "type" : "string"
                    },
                    "device_to_check": {
                        "title": "Device to check",
                        "type":"string",
                        "default": "/dev/sda"
                    }
                },
              "title" : "Target",
              "type" : "object"
          }
      },
      "inputs": {
```

```
            "ssh_string": "string",
            "ssh_key": "string",
            "device_to_check": "string",
        },
        "outputs": {
            "result": "boolean"
        }
    },
  "name" : "Checks luks is enabled on target"
}
```

### Confidenzialità del canale

```
{
  "category" : [  ],
  "driver" : "channel-confidentiality",
  "id" : 1,
  "metadata" : {
      "description" : "Evaluates SSL configuration for a given target",
      "schema" : { "config" : {
              "properties" : {
                  "host" : {
                      "title" : "Host",
                      "type" : "string"
                    },
                  "port" : {
                      "default" : 80,
                      "title" : "port",
                      "type" : "number"
                    }
                },
              "title" : "Target",
              "type" : "object"
            }
        },
        "inputs": {
            "host": "hostname",
            "port": "integer"
        },
        "outputs": {
            "result": "boolean",
            "strength": "string"
        }
    },
  "name" : "Check channel confidentiality"
}
```

## 2.5.3 Evaluation Rule

`t#1 and t#2`

```
  {
    "aer" : 1,
    "description" : "Checks storage confidentiality and network confidentiality on the
        target tufarolo.eu",
    "id" : 1,
    "mapping" : {
        "c#1":"t#1",
        "c#2":"t#2"
    },
    "name" : "Confidentiality tufarolo.eu",
    "related_ers" : [  ],
    "related_tests" : [ 1, 2 ],
    "status" : 0,
  }
```

### 2.5.4 Test

#### Confidenzialità dello storage

```
{
   "control" : 1,
   "description" : "",
   "execution_cluster" : 1,
   "id" : 11,
   "name" : "",
   "testcase" : { "config" : {
           "ssh_string" : "user@tufarolo.eu:22",
           "ssh_key" : ".....",
           "device_to_check": "/dev/sda1"
         }
     },
   "__unicode__" : "t#1"
 }
```

#### Confidenzialità del canale

```
{
   "control" : 2,
   "description" : "",
   "execution_cluster" : 1,
   "id" : 11,
   "name" : "",
   "testcase" : { "config" : {
           "host" : "tufarolo.eu",
           "port" : 80
         } },
   "__unicode__" : "t#2"
 }
```

### 2.5.5 Driver

#### Confidenzialità dello storage

```
from driver import Driver
from libraries import ssh
class StorageConfidentiality(Driver):
    def connect_to_ssh(self, inputs):
    ssh.connect(self.testinstances.get("ssh").get("ssh_string"), self.testinstances..get
        ("ssh").get("ssh_key")))

    def check_is_luks(self, inputs):
        return ssh.run("/bin/cryptsetup isLuks \%s" \% self.testinstances.get())

    def appendAtomics(self):
        self.appendAtomic(self.connect_to_ssh, lambda(x): None)
        self.appendAtomic(self.check_is_luks, lambda(x): None)
```

#### Confidenzialità del canale

```
from driver import Driver
from libraries import ssl
class ChannelConfidentiality(Driver):
    def check_ssl(self, inputs):
        target = self.testinstances.get("config").get("target")
        port = self.testinstances.get("port").get("port")
        status, strength =  ssl.check(target, port)
```

```
8           self.result.data["strength"] = strength
9           return status
10      def appendAtomics(self):
11          self.appendAtomic(self.check_ssl, lambda(x): None)
```

# Capitolo 3

# FedRAMP - Federal Risk and Authorization Management Program

## 3.1 Introduzione

In questo capitolo verrà approfondito FedRAMP, il programma federale americano per la gestione del rischio e delle autorizzazioni nella cloud. Sarà proposta un'analisi degli obiettivi del programma, specificando le problematiche in esso affrontate in relazione anche a quanto descritto nel capitolo precedente. Verrà poi esposta la struttura del documento, dopodiché ci si concentrerà sulla struttura dello stesso approfondendo i ruoli degli attori coinvolti, e il contributo che questo lavoro di tesi vuole apportare per ciascun caso trattato. In conclusione saranno esposti i concetti di *readiness* e di *compliance* al programma, e sarà approfondita l'implementazione di FedRAMP in Amazon AWS.

## 3.2 Cos'è FedRAMP

FedRAMP è il programma governativo americano l'applicazione del **FISMA** (Federal Information Security Management Act) nell'adozione di tecnologie cloud. Esso propone un approccio standardizzato al *security assessment*, alle autorizzazioni e al monitoraggio continuo di prodotti e servizi cloud, fornendo un insieme di requisiti di sicurezza e un programma di assessment indipendente, nato dalla collaborazione di esperti di sicurezza e di tecnologie cloud. Le entità coinvolte nella redazione di questo programma sono state molte: la General Services Administration (GSA), il National Institute of Standards and Technology (NIST), il dipartimento di Sicurezza Nazionale (Department of Homeland Security, DHS), il dipartimento della Difesa (Department of Defense, DOD), la National Security Agency (NSA), l'Office of Management and Budget (OMB).

I *cloud service provider* che vogliono offrire servizi per la pubblica amministrazione americana e gli uffici federali devono essere autorizzati tramite questo programma. Nonostante sia stato sviluppato nel contesto USA, la dinamicità e l'elasticità di FedRAMP ne ha permesso l'adozione *de-facto* anche in altre nazioni, specialmente dell'Asia orientale e del nord Europa.

### 3.2.1 FISMA, Federal Information Security Management Act

Il FISMA è uno standard di sicurezza, entrato in vigore come legge il 17 Dicembre 2002 come "Titolo III" dell'E-Government Act[35]: ciascun sistema che ospiti dati governativi deve essere autorizzato tramite il FISMA prima di essere messo in produzione. Esso definisce tre obiettivi principali per la sicurezza dei sistemi informativi federali:

- **Confidenzialità**, per garantire restrizioni autorizzate sull'accesso e la *disclosure* dei dati, con l'obiettivo di proteggere la privacy ed eventuale informazioni sul proprietario odegli stessi
- **Integrità**, per proteggere il dato da manipolazioni o azioni distruttive e per garantire allo stesso tempo l'autenticità e la non-repudiabilità dell'informazione
- **Disponibilità**, per assicurare condizioni di affidabilità nell'accesso al dato

A tal fine il **NIST** ha prodotto il **Federal Information Risk Management Framework**(*RMF*) il quale organizza i sistemi informatici sulla base del livello di rischio e descrive un insieme minimo di requisiti che devono essere rispettati per garantire un livello di sicurezza adeguato[36], fornendo una metodologia per la selezione dei controlli di sicurezza e per l'esecuzione del deployment e dell'assessment.

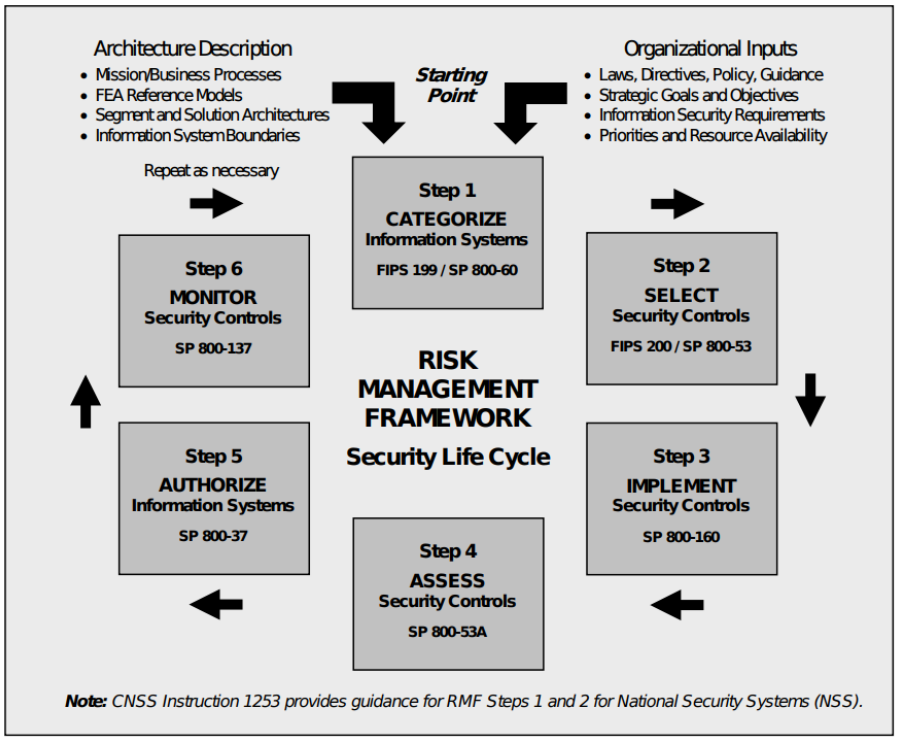



Figura 3.1: Risk Management Framework [36]

Tramite la figura 3.1 è possibile identificare le sei fasi che compongono il processo di gestione del rischio, ciclico e continuo, identificato dal framework:

- **Categorizzazione del sistema informativo**, tramite i documenti FIPS[1] 199 e NIST SP 800-60 v2[37]. Il documento FIPS 199 classifica i sistemi in base al livello di rischio, e contiene i criteri da utilizzare nella categorizzazione. Questi criteri sono basati sull'impatto potenziale di una violazione delle proprietà di confidenzialità, integrità e disponibilità sul sistema e sono: i) Rischio basso, impatto limitato, ii) Rischio medio, con serie conseguenze, iii) Rischio elevato con conseguenze gravi e catrastofiche. FIPS 199 si applica a tutti i sistemi, ad esclusione di quelli designati per l'uso della sicurezza nazionale.

- **Selezione dei controlli di sicurezza**, mediante i documenti FIPS 200 e SP 800-53. FIPS 200 fornisce i requisiti di sicurezza minimi (e i relativi controlli) per ogni categoria definita nel FIPS 199. Il documento NIST 800-53[36] invece definisce i controlli di sicurezza e fornisce le linee guida per scegliere i profili da impiegare per soddisfare i requisiti minimi di sicurezza, in base all'impatto del sistema. I controlli di sicurezza sono divisi in 17 famiglie e vengono divisi in tre classi (controlli di gestione, controlli operazionali e controlli tecnici).

- **Implementazione dei controlli**, guidata dal documento SP 800-160[38]

- **Assessment dei controlli di sicurezza**, regolato dal documento SP 800-53[36]

- **Autorizzazione del sistema informativo**, basata sul documento SP 800-37[39]

- **Fase di monitoraggio**[40]

### 3.2.2 Obiettivo di FedRAMP

L'obiettivo di FedRAMP è quello di fornire un framework per semplificare il processo di autorizzazione dei servizi cloud adottati dagli enti governativi USA. Prima dell'adozione di questo programma i produttori di sistemi e applicativi dovevano eseguire l'intero processo di autorizzazione per ciascuna delle agenzie che adottasse il sistema, così come ogni ente gestiva un processo di gestione del rischio a sé stante, anche nel caso in cui un'altra agenzia avesse già adottato servizi e misure di sicurezza analoghi. FedRAMP affronta la problematica nei seguenti modi[41]:

- Fornendo processi di valutazione della sicurezza e autorizzazione congiunti, basati su una serie di requisiti e controlli standardizzati, sulla base dell'impatto del sistema

- Offrendo un programma di analisi della conformità in grado di produrre

[1] Federal Information Processing Standards

- Strutturando un processo di analisi e valutazione della conformità condotto da Organizzazioni di terze parti approvate (3PAO), per valutare costantemente la capacità di un provider di servizi cloud (CSP) di soddisfare requisiti di sicurezza desiderati
- Coordinando servizi di monitoraggio continuo
- Fornendo pacchetti di autorizzazione composti da servizi cloud già revisionati da una Joint Authorization Board (JAB), composta da esperti di sicurezza provienienti dal DHS, dalla GSA e del DoD.
- Offrendo un linguaggio standardizzato per aiutare i dipartimenti e gli enti governativi ad integrare i requisiti di FedRAMP all'interno dei processi interni
- Un repository di pacchetti di autorizzazione per i servizi cloud che possono essere utilizzati dal governo

L'utilizzo di un framework centralizzato ha inevitabilmente determinato un risparmio notevole anche in termini economici, sia per il governo americano, che per i cloud service provider: si stima che questo ammonti a circa $250,000 per ciascun sistema autorizzato, per un totale di circa 160 implementazioni dello standard FISMA. Il risparmio stimato per il governo è quindi di 40 milioni di dollari [42].

## 3.3 Struttura

Il programma fornisce un percorso che i fornitori di servizi cloud possono intraprendere per ottenere una autorizzazione provvisoria, da sottoporre in una successiva fase di *security assessment* che verrà poi revisionata dalla *JAB*. Mediante questo approccio preventivo è possibile quindi anticipare l'*assessment* dei controlli di sicurezza e di velocizzare il processo di autorizzazione definitiva dell'applicativo o sistema cloud.

### 3.3.1 CSP: FedRAMP readiness

Il provider che vuole partecipare a FedRAMP deve innanzitutto soddisfare una serie di requisiti di *readiness*:

1. Essere in grado di trattare eventuali processi forensi elettronici e contenziosi
2. Essere in grado di definire e descrivere chiaramente i confini del proprio sistema
3. Identificare le responsabilità del cliente e le azioni che questo deve compiere per implementare i controlli di sicurezza
4. Fornire un meccanismo di identificazione e autenticazione a due fattori per l'accesso via rete agli account privilegiati

5. Fornire un meccanismo di identificazione e autenticazione a due fattori per l'accesso via rete agli account non privilegiati

6. Fornire un meccanismo di identificazione e autenticazione a due fattori per l'accesso locale agli account privilegiati

7. Avere la possibilità di eseguire analisi del codice per le soluzioni software proprietarie

8. Avere protezioni di confine garantendo isolamento logico e fisico degli asset

9. Avere l'abilità di rimediare a situazioni di rischio elevato entro i 30 giorni (90 giorni per le situazioni di rischio moderato)

10. Fornire un inventario e configurazioni standard per tutti i dispositivi

11. Avere meccanismi di sicurezza che impediscano la fuoriuscita di informazioni nell'utilizzo di mezzi di comunicazione condivisi

12. Adottare meccanismi di crittografia per preservare la confidenzialità e l'integrità dei dati trasmessi sulla rete

Se queste condizioni sono rispettate, è possibile sottomettere un modulo di richiesta di ammissione al programma, disponibile online sul sito web di FedRAMP. A questo seguirà una notifica automatica all'ufficio di gestione del programma e alla *Joint Advisory Board*.

Oltre alle finalità precedentemente esposte, il prodotto sviluppato in questo lavoro di tesi punta a supportare il processo di verifica dei requisiti richiesti per la *readiness*. Poiché questi sono essenzialmente di carattere procedurale, saranno implementati sotto forma di questionario. Per i punti 11 e 12 possono essere forniti anche controlli tecnici, legati però alle tecnologie effettivamente implementate.

### 3.3.2 Processo di autorizzazione

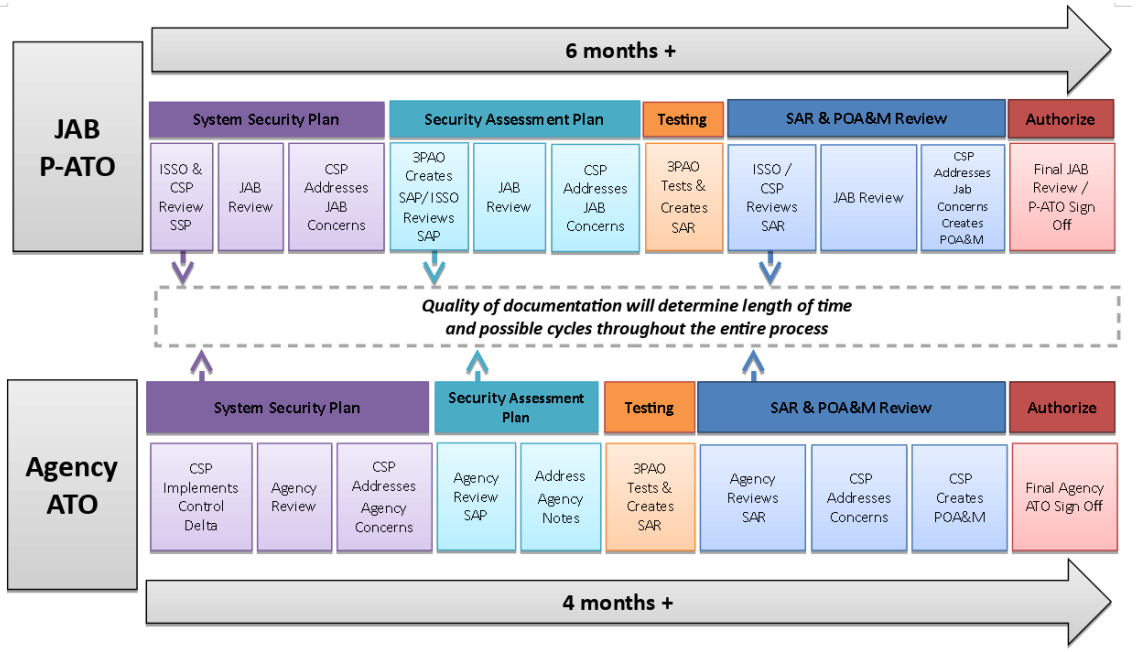



Figura 3.2: Processo di autorizzazione [41]

Nel modulo di richiesta di ammissione al programma il provider fornisce informazioni sul proprio sistema, categorizzandolo sulla base delle direttive contenute nel documento NIST SP 800-60 V2[37], e decide la *baseline* dei controlli di sicurezza da implementare in base alla sensibilità del sistema (bassa o media). A seguito di una revisione della *readiness* da parte dell'ufficio di gestione del programma, sarà possibile avviare il processo di richiesta dell'autorizzazione provisoria (P-ATO): il *cloud service provider* deve quindi ricercare ed assumere un'organizzazione di terze parti, tra quelle specificate sul sito di FedRAMP. L'attore principale è il fornitore di servizi, il quale sottometterà un *security package* e sarà designato come candidato all'autorizzazione. Lo svolgimento del processo può avvenire in due modi:

- *Processo condotto da un'agenzia federale*, che vuole far autorizzare a FedRAMP i sistemi cloud correntemente attivi. In tal caso il processo è totalmente controllato e monitorato dall'ente, il cui compito è quello di sorvegliare sull'implementazione dei controlli di sicurezza mancanti nei sistemi del fornitore di servizi. L'autorizzazione sarà quindi emessa direttamente dall'agenzia.
- *Processo mantenuto dalla Joint Authorization Board*, che analizzerà il livello di sicurezza del fornitore ed emetterà una autorizzazione provvisoria

### Attori coinvolti nel processo di autorizzazione

Gli attori coinvolti sono quindi gli enti federali che desiderano adottare un servizio cloud, i fornitori del servizio, e le organizzazioni di terze parti che ne effettuano l'analisi della sicurezza.

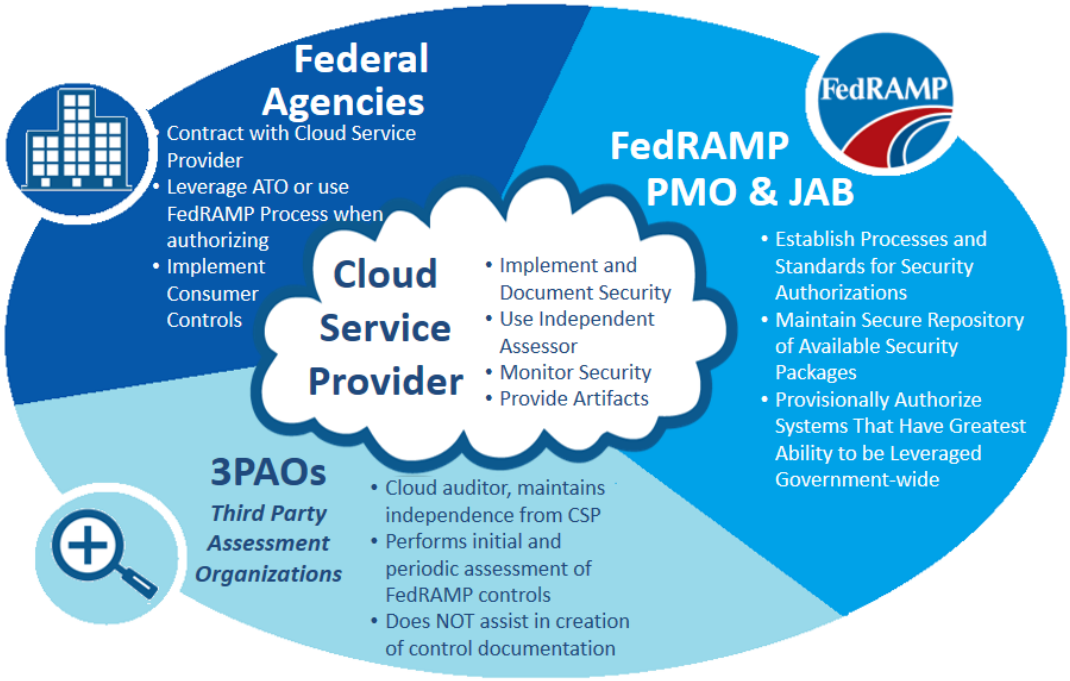



Figura 3.3: Attori coinvolti nel processo di autorizzazione [41]

### Enti federali

Il ruolo degli enti federali è quello di garantire che, per tutti i progetti nei quali sono coinvolte tecnologie cloud, siano rispettati i requisiti FedRAMP, venga effettuato l'assessment dei controlli di base, e siano stati forniti i template corretti.

Le agenzie devono catalogare i propri sistemi in un inventario, specificando quali di questi siano di tipo cloud e quali invece siano sistemi tradizionali. È raccomandabile avere un referente che sia preparato a rispondere a quesiti riguardanti l'implementazione dei requisiti FedRAMP. Per facilitare ciò alcune informazioni utili potrebbero essere *i) il nome del sistema cloud, ii) la descrizione del servizio fornito dal sistema, iii) il contatto dl proprietario del sistema, iv) la data di autorizzazione, v) lo stato della compliance*[43].

In fase di migrazione di sistemi tradizionali sul cloud, così come nell'adattamento di servizi cloud pre-esistenti a FedRAMP, è necessario che i requisiti del programma siano rispettati. Le agenzie devono quindi effetturare un'analisi approfondita delle conseguenze del cambio di tecnologia, per determinare i controlli di sicurezza aggiuntivi da effettuare. Analogamente, gli stessi controlli di sicurezza devono interessare eventuali sistemi cloud installati ed utilizzati internamente (private-cloud). In tal caso è necessario predisporre un'analisi condotta da terze parti accreditate. Se, invece, il fornitore del servizio è un privato e questi non ha effettuato l'assessment dei controlli di sicurezza FedRAMP, l'en-

te governativo è tenuto ad informare il provider e richiederne l'adeguamento immediato.

Le agenzie, sulla base di specifiche esigenze, possono sottomettere eventuali controlli aggiuntivi rispetto a quelli basilari; questi devono devono essere adeguatamente documentati e motivati. Gli altri enti possono poi decidere di adottare questi controlli. Allo stesso modo, come già trattato, le agenzie possono riutilizzare autorizzazioni emesse per altri enti.

Sulla base del E-Government Act del 2002 (Titolo III, Sezione 3544), le agenzie devono inoltre effettuare analisi del rischio periodiche, valutando l'impatto di eventuali violazione della confidenzialitàe dell'integrità dei dati[43].

Uno degli obiettivi della piattaforma presentata in questo lavoro di tesi, è quello di fornire uno strumento a supporto delle agenzie federali per la gestione della sicurezza degli asset inventariati (sistemi tradizionali e cloud), fornendo sia meccanismi di *auto-discovery* degli stessi (ad esempio mediante l'integrazione con le API dei cloud service provider, oppure tramite la scansione delle reti locali), sia un processo di monitoraggio continuativo della stato della compliance.

#### Organizzazioni di terze parti (3PAO)

Affinché un servizio cloud possa essere autorizzato da FedRAMP, è necessario che un'organizzazione di terze parti ne analizzi la sicurezza mediante l'esecuzione dei controlli standardizzati nel documento NIST 800-53. Per ottenere l'abilitazione ad effettuare queste analisi, l'organizzazione può decidere di iniziare un percorso di accreditamento, il cui obiettivo è quello di garantire che che le analisi siano effettuate in maniera consistente, dettagliata e indipendente. A tal fine, l'organizzazione deve inviare sottomettere del materiale in grado di dimostrare di essere in grado sia di eseguire analisi tecniche adeguate ai livelli attesi, sia di avere competenza nella gestione dei processi di *compliance*. Questi criteri vengono certificati dalla *American Association for Laboratory Accreditation (A2LA)* che, dopo aver verificato le effettive competenze tecniche in capo alll'organizzazione, effettua un processo di controllo della conformità rispetto allo standard ISO/IEC 17020 (Disposizioni per la transizione degli accreditamenti degli Organismi di ispezione (OdI)).

Il testing della sicurezza nei confronti dei fornitori di servizi deve essere effettuato in modo equo, tramite controlli scelti sulla base della loro categoria di sensibilità. La periodicità è annuale, e la stessa organizzazione non può effettuare una scansione per due anni consecutivi. In alcuni casi può essere necessario eseguire test automatici come utenti autenticati e con pieni privilegi, in modo da poter determinare con precisione le vulnerabilità e il relativo impatto sul sistema. Solo in questo modo, infatti, è possibile avere una visione globale del sistema (ad es. accedere al registro di sistema di Windows, agli attributi dei file di sistema, ai pacchetti e alle patch effettivamente installate). L'utilizzo di un utente con privilegi limitati può restituire sia falsi positivi (ad esempio nel caso di un test di scrittura su directory interne al sistema eseguito in un ambiente *chroot*) e falsi negativi (in caso di assunzioni derivate dall'impossibilità per l'utente di leggere determinati parametri). Eventuali analisi del codice, invece, sono demandate al *cloud service provider*.

Per guidare questo processo in modo standard, FedRAMP fornisce vari template, scaricabili dal sito ufficiale:

- *Security Assessment Plan Template*, il cui scopo è quello di descrivere il piano per l'analisi della sicurezza. Prima di redigere questo documento, l'organizzazione deve incontrarsi col fornitore di servizi cloud per discutere i test da eseguire. Eventuale supporto può essere erogato dall'*Information System Security Officier* di FedRAMP.

- *Security Assessment Test Cases*, in cui vengono descritti i casi di test sulla base del documento NIST 800-53A; alcuni di questi, tuttavia, differiscono poiché sono stati adattati al contesto *cloud*. Nel caso in cui l'organizzazione debba implementare versioni alternative dei controlli di sicurezza, è necessario che i casi di test vengano scritti in modo idoneo ad attestarne l'efficacia.

- *Security Assessment Report*, assiste la parte di reportistica, il cui obiettivo è quello di dettagliare l'analisi eseguita sui sistemi del fornitore di servizi cloud, riportando le evidenze trovate, le possibili operazioni di mitigazione e le eventuali raccomandazioni.

Il sistema progettato nell'ambito di questo progetto di tesi, mira a diventare uno strumento di supporto delle organizzazioni di terze parti. Uno dei possibili sviluppi in tal senso, potrebbe consistere nell'integrazione dei template presentati in questa sezione all'interno della piattaforma realizzata, in modo da poter guidare l'intero processo di *security assessment*, e consentire anche ad organizzazioni differenti di mantenere una cronologia delle analisi svolte su uno specifico servizio cloud.

#### Fornitori di servizi cloud

Il *cloud service provider* può essere sia un'entità di terze parti commerciale che un altro ente governativo od agenzia. La sua responsabilità è quella di implementare i controlli di sicurezza, di assumere un'organizzazione di terze parti indipendente che effettui l'assessment annuale e di effettuare tutte le procedure per la creazione e la manutenzione delle proprie autorizzazioni.

Le modalità con cui un fornitore di servizi può essere autorizzato sono tre:

- Il provider può inviare la documentazione appropriata al PMO (ufficio di gestione del programma FedRAMP) e alla Joint Advisory Board, che possono erogare un'autorizzazione provvisoria (P-ATO, Provisional Authorization to Operate)

- Il provider può inviare la documentazione appropriata al PMO e ad un'agenzia, che può erogare un'autorizzazione ad operare (ATO, Authorization to Operate). Come già spiegato, un'altra agenzia può utilizzare poi la stessa autorizzazione, abbreviando i tempi di approvazione.

- Il provider può inviare la documentazione per intraprendere autonomamente un percorso di tipo "CSP supplied". In tal caso, dovrà assumere una organizzazione di terze parti che ne analizzi la sicurezza.

Quindi, affinché un sistema cloud sia conforme con FedRAMP:

- deve essere stato creato e sottomesso un pacchetto, utilizzando i template idonei
- deve essere stato eseguito l'assessment, da parte di un'organizzazione di terze parti accreditata e indipendente, mediante l'esecuzione dei controlli di sicurezza relativi al livello di sensibilità del sistema (basso o medio, in quanto i sistemi ad alta sensibilità non sono supportati dal programma e devono essere gestiti separatamente).
- l'assessment deve aver restituito risultati positivi, attestando che i requisiti di sicurezza siano effettivamente verificati
- deve essere stata erogata un'autorizzazione ad operare, provvisoria o definitiva

## 3.4 CSP: Ulteriori oneri

Successivamente al rilascio dell'autorizzazione provvisoria ad operare, il *cloud service provider* deve sottomettere altri documenti la cui redazione è, ancora una volta, guidata da template disponibili sul sito. Questi devono poi essere inclusi in un *security package*.

- **FIPS 199** nel quale, come precedentemente illustrato, va effettuata la categorizzazione dei sistemi in base al relativo livello di sensibilità, sulla base del documento NIST 800-60. I fornitori di servizi a livello IaaS e PaaS devono seguire le indicazioni della sezione C.3.5 di tale documento.
- **E-Authentication** per l'analisi dei processi dell'autenticazione elettronica, al fine di garantire che siano state implementate misure idonee a minimizzare il livello di rischio. I criteri da utilizzare in questo caso, sono riferiti al documento NIST 800-63.
- **PTA e PIA**, Privacy Threshold Analysis & Privacy Impact Assessment, composti da solo quattro domande che hanno lo scopo di individuare il livello di privacy relativo al sistema e di pianificare l'esecuzione degli eventuali controlli di sicurezza relativi In particolare va specificato se l'esecuzione di questi controlli implica l'utilizzo di eventuali meccanismi di autenticazione con impatti sulla privacy (ad esempio utilizzo dell'autenticazione biometrica per l'accesso ai locali) per gli operatori dell'organizzazione di terze parti che eseguono gli stessi.
- **CTW**, Control Taylor WorkBook, il cui obiettivo è quello di riassumere gli scenari di utilizzo dei servizi offerti da parte dell'agenzia.
- **CIS**, Control Implementation Summary, nel quale è indicato lo stato dell'implementazione dei controlli nel sistema e il responsabile del processo di gestione degli stessi.

- **SSP**, System Security Plan, il quale descrive le modalità con cui sono (o saranno) implementati i controlli di sicurezza previsti.

Inoltre il fornitore del servizio deve fornire un manuale utente che spieghi le modalità di utilizzo del sistema (ad esempio illustrando fornendo la documentazione per eventuali Dashboard o interfacce API).

Il ruolo del prodotto presentato nel lavoro di tesi, in tal caso, è quello di supportare il processo di sottomissione questi documenti, fornendo un framework per implementare il template relativo a ciascuno di essi.

## 3.5 System Security Plan

Nel System Security Plan sono descritte le implementazioni dei controlli di sicurezza in relazione all'architettura del sistema del provider. Il sistema, infrastruttura, piattaforma o applicazione che sia, è in tal caso trattato come un insieme di componenti, ognuno dei quali deve essere documentato, utilizzando terminologie chiare, non ambigue, ed aderenti il più possibile alla nomenclatura comunemente utilizzata nell'ambito di interesse del componente stesso. In particolare, dovranno essere discussi gli aspetti di virtualizzazione e di conservazione dei dati, focalizzandosi sulla determinazione dei confini determinati dall'utilizzo delle varie tecnologie.

### 3.5.1 Virtualizzazione dei nodi di calcolo

L'ambito di virtualizzazione è gestito dalla sezione 9 del SSP. Bisogna chiarire:

- quali componenti fanno parte del sistema fisico
- quali componenti e quali funzionalità sono parte del sistema virtualizzato o astratto

Questa suddivisione si rispecchia immediatamente nella fase di categorizzazione degli *asset*. Bisogna infatti suddividere gli eventuali sistemi in:

- sistemi standard, installati sugli host
- sistemi virtualizzati, ospiti di un *hypervisor*
- sistemi che virtualizzano, ovvero gli *hypervisor*, per cui bisogna specificare se la virtualizzazione avviene in modalità:
  - bare-metal (in cui la componente hardware è gestita direttamente dal sistema virtuale), anche detta virtualizzazione di tipo 1
  - host-based (ovvero confinata su un server, che gestisce la parte hardware), nota come virtualizzazione di tipo 2

Per i provider *IaaS* è necessario anche specificare le modalità di installazione dei sistemi virtuali. Infatti, in caso di infrastrutture cloud *full-managed*, è il cloud service provider che installa e configura i sistemi operativi, garantendo poi l'accesso

al cliente (agenzia governativa); al contrario, in infrastrutture *self-managed*, è il cliente che, tramite una dashboard, effettua autonomamente il *deploy* del sistema virtuale. A questo punto bisogna distinguere ulteriori casistiche:

- Il fornitore di servizi mette a disposizione delle immagini di base, con software preinstallato
- Il cliente installa autonomamente il sistema operativo, caricando un'immagine di base, o l'immagine di un sistema pre-esistente, sull'infrastruttura del provider

Inoltre occorre specificare eventuali meccanismi di controllo degli accessi per la gestione delle risorse, l'eventuale modalità di gestione della memoria condivisa, e le tecnologie implementate per la resilienza dell'infrastruttura.

### 3.5.2 Virtualizzazione della rete

Grazie alle tecnologie di *software-defined networking* è possibile effettuare anche la virtualizzazione dello stack e dei componenti di rete, come switch e router. Questi eventualmente possono essere integrati con appliance fisiche che supportino protocolli SDN (*OpenFlow, FabricPath*); è necessario perciò specificare:

- quali componenti di rete siano virtualizzati
- quali invece siano apparati fisici

L'eventuale adozione di meccanismi di segregazione, come ad esempio le VLAN per il partizionamento del livello 2 dello Stack ISO/OSI, le VLAN estese, gli switch virtuali distribuiti, i gateway di sicurezza virtuali, introduce ulteriori parametri da valutare e da specificare nel System Security Plan, in particolare per la determinazione dei confini (approfondita nel prossimo paragrafo). Innanzitutto è necessario fornire un diagramma che illustri dettagliatamente la direzione del flusso dei dati sulla rete, indipendentemente da come la topologia della stessa sia strutturata, ed evidenziando gli eventuali componenti critici della topologia che determinano il flusso stesso.

### 3.5.3 Determinazione dei confini e controlli di sicurezza relativi

Per determinare i confini, occorre specificare in modo chiaro ed articolato dove il layer costituito dal servizio cloud inizia e dove finisce. Ciò contribuisce alla risoluzione delle problematiche relative alla *shared responsability*: se un servizio *SaaS* che vuole essere autorizzato a FedRAMP utilizza un servizio *PaaS*, è necessario effettuare l'assessment e l'autorizzazione anche del servizio *PaaS*. Analogamente, nel caso di una *PaaS* ospitata su una *IaaS*, è importante specificare il livello a cui sono implementati i requisiti di sicurezza desiderati e gestire l'assessment in modo opportuno.

Alcune domande, relative perlopiù agli aspetti di rete e di storage, consigliate dal documento Guide to Understanding FedRAMP per la determinazione dei confini sono:

- Viene effettuato isolamento al livello fisico, mediante l'utilizzo di interfacce di rete dedicate?
- Viene effettuata segregazione a livello 2 dello stack ISO/OSI?
- Sono implementate delle VLAN? Qual'è la definizione di tenant utilizzata?
- Eventuali VLAN rispecchiano la separazione dei tenant, oppure tenant diversi possono avere accesso alla stessa VLAN?
- Viene effettuato del monitoraggio per prevenire il VLAN-hopping?
- Viene effettuato bonding sulle interfacce di rete per migliorarne prestazioni e affidabilità?
- Sono presenti ACL per fornire isolamento tra i tenant?
- È implementato il protocollo IPSec per definire i confini tra reti di tenant diversi?
- Sono definiti dei confini a livello geografico sia per i dati in transito che per quelli memorizzati?
- È possibile, per i clienti, conoscere la locazione geografica dei propri dati?
- Il sistema utilizza tecnologie DAS[2], NAS[3], o SAN[4]?
- Se il sistema ha una SAN, è connessa tramite fibra ottica o iSCSI?

Un ulteriore fattore da considerare è determinato dalle funzionalità di migrazione diretta (*live migrations*) impiegate. Ciò riguarda principalmente i fornitori di servizi *IaaS* e *PaaS* che, per ragioni di prestazioni e ridondanza, possono spostare geograficamente le risorse cloud, come macchine virtuali e container. È fondamentale, infatti, che in tali casi i dati persistano all'interno del confine designato, sia che siano memorizzati su uno storage, sia che siano in transito su una rete: gli indirizzi IP dichiarati all'interno del confine devono rimanere tali, così come il traffico di rete non deve essere veicolato attraverso regioni geografiche non dichiarate. Occorre perciò specificare se le migrazioni dirette siano fatte in modo programmato ed automatizzato, oppure in modo manuale e, nel primo caso, dichiarare le regole utilizzate per gestirle.

[2]Direct Attached Storage
[3]Network Attached Storage
[4]Storage Area Network

## 3.6 Controlli di sicurezza per la conformità - NIST 800-53 e FedRAMP

### 3.6.1 Categorie dei controlli

I controlli di sicurezza di FedRAMP sono basati sul documento NIST 800-53. Alcuni, di carattere essenzialmente procedurale, sono stati introdotti da FedRAMP, per altri sono stati proposti miglioramenti. La catalogazione dei controlli è effettuata sulla base del livello di sensibilità (basso e moderato); essi sono inoltre suddivisi in famiglie, ciascuna delle quali è contrassegnata univocamente da un *ID*[44]. Ogni famiglia appartiene a una classe (Tecnico, Operativo, Management), che ne identifica l'ontologia. Nella tabella 3.1, è proposto un conteggio dei controlli di sicurezza in ogni famiglia, per le *baseline* relative ai livelli di sensibilità basso e moderato.

## 3.7 FedRAMP in Amazon Web Services e Azure

*Amazon* e *Microsoft* sono due esemi di fornitori di servizi aver seguito il programma di autorizzazione FedRAMP, soddisfacendo i controlli di sicurezza previsti tramite i modelli illustrati in questo capitolo.

Il servizio AWS che garantisce la compliance FedRAMP prende il nome di *GovCloud*, ed ha ricevuto un'autorizzazione P-ATO e diverse ATO per il livello di sensibilità più elevato. Inoltre anche altri servizi AWS, appartenenti alle regioni *Stati Uniti occidentali* e *Stati Uniti orientali* sono risultati conformi al programma FedRAMP, ottenendo autorizzazioni ATO per il livello di sensibilità *moderato*.

Per quanto riguarda Microsoft invece, è il servizio *Azure* (e Azure per enti pubblici) ad aver ricevuto una P-ATO per il livello di sensibilità *moderato*. Inoltre, il servizio *Microsoft Dynamics CRM Online per enti pubblici* ha ricevuto una ATO dall'HUD mentre Microsoft Office 365 per il Governo degli Stati Uniti ha ricevuto una ATO dal DHHS (Dipartimento della Salute e dei Servizi Umani).

| **ID** | **Famiglia** | **Classe** | **Livello Basso** | **Livello moderato** |
|---|---|---|---|---|
| AC | Controllo degli accessi | Tecnico | 11 | 18 |
| AT | Awareness & training | Operativo | 4 | 4 |
| AU | Audit and accountability | Tecnico | 10 | 11 |
| CA | Certification, Accdeditation & Security Assessment | Management | 7 | 8 |
| CM | Configuration Management | Operativo | 8 | 11 |
| CP | Contingency Planning | Operativo | 6 | 9 |
| IA | Identification and authentication | Tecnico | 7 | 8 |
| IR | Incident response | Operativo | 7 | 9 |
| MA | Maintenance | Operativo | 4 | 6 |
| MP | Media protection | Operativo | 4 | 7 |
| PE | Physical and environmental protection | Operativo | 10 | 16 |
| PL | Planning | Management | 3 | 4 |
| PS | Personnel security | Operativo | 8 | 8 |
| RA | Risk assessment | Management | 4 | 4 |
| SA | System and services acquisition | Management | 6 | 9 |
| SC | System and communications protection | Tecnico | 10 | 20 |
| SI | System and information integrity | Operativo | 7 | 12 |
| **TOT** | - | - | **116** | **164** |

Tabella 3.1: Famiglie di controlli[45][46]

# Capitolo 4

# Implementazione dei controlli di sicurezza FedRAMP in Moon Cloud

## 4.1 Introduzione

In questo capitolo verrà effettuata un'analisi dei controlli di sicurezza elencati nel capitolo precedente, classificandoli in *controlli automatici* e *controlli procedurali*. Dopodiché verra offerta una possibile implementazione per ciascuna delle due tipologie di controlli, i quali verranno integrati in Moon Cloud. Per i controlli automatici sarà utilizzato Open Scap, uno strumento per il l'auditing realizzato da Red Hat e certificato dal NIST. I controlli procedurali invece, eseguono l'*processi di business* e vanno ad indirizzare tutte quelle proprietà di carattere puramente qualitativo per cui è fondamentale l'interazione umana. Questi saranno implementati con un *driver Moon Cloud ad interazione umana*, che somministra un questionario online ad un target.

## 4.2 Analisi dei controlli di sicurezza

### 4.2.1 Access Control

## 4.3 Implementazione dei controlli automatici

### 4.3.1 Il framework OpenSCAP

### 4.3.2 Driver OpenSCAP per Moon Cloud

### 4.3.3 OpenSCAP per la FedRAMP readiness

### 4.3.4 OpenSCAP per la NIST 800-53

## 4.4 Controlli ad interazione umana

### 4.4.1 Questionari per l'assessment dei controlli procedurali

### 4.4.2 Templating del questionario

# Capitolo 5

# Validazione del framework

## 5.1 Deployment di Moon Cloud

## 5.2 Sicurezza del deployment

### 5.2.1 Caratteristiche del deployment

### 5.2.2 Confidenzialità

### 5.2.3 Integrità

### 5.2.4 Disponibilità e affidabilità

### 5.2.5 Data remainance

## 5.3 Scalabilità e Prestazioni

### 5.3.1 una sezione per ogni security control eseguito

# Capitolo 6

# Conclusioni e sviluppi futuri

# Bibliografia


[1] National Institute of Standards e Technology (Peter Mell Timothy Grance). *The NIST Definition of Cloud Computing*. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[2] S. Dhar. «From outsourcing to Cloud computing: Evolution of IT services». In: Technology Management Conference (ITMC), 2011 IEEE International. IEEE.

[3] Claudio A. Ardagna et al. «From Security to Assurance in the Cloud: A Survey». In: *ACM Comput. Surv.* 48.1 (lug. 2015), 2:1–2:50. ISSN: 0360-0300. DOI: 10.1145/2767005. URL: http://doi.acm.org/10.1145/2767005.

[4] N. Gruschka e L. L. Iacono. «Vulnerable Cloud: SOAP Message Security Validation Revisited». In: *2009 IEEE International Conference on Web Services*. Lug. 2009, pp. 625–631. DOI: 10.1109/ICWS.2009.70.

[5] Sven Bugiel et al. «AmazonIA: When Elasticity Snaps Back». In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS '11. Chicago, Illinois, USA: ACM, 2011, pp. 389–400. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046753. URL: http://doi.acm.org/10.1145/2046707.2046753.

[6] Thomas Ristenpart et al. «Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds». In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS '09. Chicago, Illinois, USA: ACM, 2009, pp. 199–212. ISBN: 978-1-60558-894-0. DOI: 10.1145/1653662.1653687. URL: http://doi.acm.org/10.1145/1653662.1653687.

[7] M. Green. «The Threat in the Cloud». In: *IEEE Security Privacy* 11.1 (gen. 2013), pp. 86–89. ISSN: 1540-7993. DOI: 10.1109/MSP.2013.20.

[8] Huan Liu. «A New Form of DOS Attack in a Cloud and Its Avoidance Mechanism». In: *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*. CCSW '10. Chicago, Illinois, USA: ACM, 2010, pp. 65–76. ISBN: 978-1-4503-0089-6. DOI: 10.1145/1866835.1866849. URL: http://doi.acm.org/10.1145/1866835.1866849.

[9] F. Rocha e M. Correia. «Lucy in the sky without diamonds: Stealing confidential data in the cloud». In: *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*. Giu. 2011, pp. 129–134. DOI: 10.1109/DSNW.2011.5958798.

[10] Sören Bleikertz et al. «Client-controlled Cryptography-as-a-service in the Cloud». In: *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*. ACNS'13. Banff, AB, Canada: Springer-Verlag, 2013, pp. 19–36. ISBN: 978-3-642-38979-5. DOI: 10.1007/978-3-642-38980-1_2. URL: http://dx.doi.org/10.1007/978-3-642-38980-1_2.

[11] S. De Capitani di Vimercati et al. «Three-server swapping for access confidentiality». In: *IEEE Transactions on Cloud Computing* PP.99 (2015), pp. 1–1. ISSN: 2168-7161. DOI: 10.1109/TCC.2015.2449993.

[12] S. De Capitani di Vimercati et al. «Integrity for join queries in the cloud». In: *IEEE Transactions on Cloud Computing* 1.2 (lug. 2013), pp. 187–200. ISSN: 2168-7161. DOI: 10.1109/TCC.2013.18.

[13] S. A. Almulla e Chan Yeob Yeun. «Cloud computing security management». In: *2010 Second International Conference on Engineering System Management and Applications*. Mar. 2010, pp. 1–7.

[14] Hassan Takabi e James B.D. Joshi. «Policy Management as a Service: An Approach to Manage Policy Heterogeneity in Cloud Computing Environment». In: *45th Hawaii International Conference on System Sciences (HICSS-45)*. IEEE, 2012, pp. 5500–5508. URL: http://d-scholarship.pitt.edu/13526/.

[15] Philogene A. Boampong e Luay A. Wahsheh. «Different Facets of Security in the Cloud». In: *Proceedings of the 15th Communications and Networking Simulation Symposium*. CNS '12. Orlando, Florida: Society for Computer Simulation International, 2012, 5:1–5:7. ISBN: 978-1-61839-785-0. URL: http://dl.acm.org/citation.cfm?id=2331762.2331767.

[16] F. John Krautheim. «Private Virtual Infrastructure for Cloud Computing». In: *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*. HotCloud'09. San Diego, California: USENIX Association, 2009. URL: http://dl.acm.org/citation.cfm?id=1855533.1855538.

[17] Stefan Berger et al. «vTPM: Virtualizing the Trusted Platform Module». In: *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*. USENIX-SS'06. Vancouver, B.C., Canada: USENIX Association, 2006. URL: http://dl.acm.org/citation.cfm?id=1267336.1267357.

[18] Michael Velten e Frederic Stumpf. «Secure and Privacy-Aware Multiplexing of Hardware-Protected TPM Integrity Measurements among Virtual Machines». In: *Information Security and Cryptology – ICISC 2012: 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers*. A cura di Taekyoung Kwon, Mun-Kyu Lee e Daesung Kwon. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 324–336. ISBN: 978-3-642-37682-5. DOI: 10.1007/978-3-642-37682-5_23. URL: http://dx.doi.org/10.1007/978-3-642-37682-5_23.

[19] Jakub Szefer e Ruby B. Lee. «Hardware-Enhanced Security for Cloud». In: *Secure Cloud Computing*. Berlin: Springer, 2014, pp. 57–76. URL: http://link.springer.com/chapter/10.1007%2F978-1-4614-9278-8_3.

[20] C. A. Ardagna et al. «On the Management of Cloud Non-Functional Properties: The Cloud Transparency Toolkit». In: *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. Mar. 2014, pp. 1–4. DOI: 10.1109/NTMS.2014.6814039.

[21] K.M. Goertzel et al. *Software Security Assurance: A State-of-the Art Report (SOAR)*. Information Assurance Technology Analysis Center, 2007. URL: https://books.google.it/books?id=xxHPMgEACAAJ.

[22] Iain MacNeil e Xiao Li. «"Comply or Explain": market discipline and non-compliance with the Combined Code». In: *Corporate Governance: An International Review* 14.5 (2006), pp. 486–496. URL: http://EconPapers.repec.org/RePEc:bla:corgov:v:14:y:2006:i:5:p:486-496.

[23] L. M. Riungu, O. Taipale e K. Smolander. «Research Issues for Software Testing in the Cloud». In: *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. Nov. 2010, pp. 557–564. DOI: 10.1109/CloudCom.2010.58.

[24] Z. Chiba et al. «A survey of intrusion detection systems for cloud computing environment». In: *2016 International Conference on Engineering MIS (ICEMIS)*. Set. 2016, pp. 1–13. DOI: 10.1109/ICEMIS.2016.7745295.

[25] M. Ficco, L. Tasquier e R. Aversa. «Intrusion Detection in Cloud Computing». In: *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Ott. 2013, pp. 276–283. DOI: 10.1109/3PGCIC.2013.47.

[26] E. et al. Damiani. *Cumulus. D2.3 Certification models v.2*. Seventh Framework Programme - CUMULUS FP7, 2014.

[27] Marco Anisetti et al. «A Test-based Security Certification Scheme for Web Services». In: *ACM Trans. Web* 7.2 (mag. 2013), 5:1–5:41. ISSN: 1559-1131. DOI: 10.1145/2460383.2460384. URL: http://doi.acm.org/10.1145/2460383.2460384.

[28] E. Damiani et al. «WS-Certificate». In: *2009 Congress on Services - I*. Lug. 2009, pp. 637–644. DOI: 10.1109/SERVICES-I.2009.132.

[29] M. Anisetti, C. Ardagna e E. Damiani. «2012». In: (Low-Cost Security Certification Scheme for Evolving Services).

[30] Ernesto Damiani Marco Anisetti Claudio Ardagna. «A Test-Based Security Certification Scheme for Web Services». In: *ACM Transactions on the Web (TWEB)* (2013).

[31] M. Anisetti, C. A. Ardagna e E. Damiani. «Security Certification of Composite Services: A Test-Based Approach». In: *2013 IEEE 20th International Conference on Web Services*. Giu. 2013, pp. 475–482. DOI: 10.1109/ICWS.2013.70.

[32] G. Spanoudakis, E. Damiani e A. Maña. «Certifying Services in Cloud: The Case for a Hybrid, Incremental and Multi-layer Approach». In: *2012 IEEE 14th International Symposium on High-Assurance Systems Engineering*. Ott. 2012, pp. 175–176. DOI: 10.1109/HASE.2012.16.

[33] N. S. Chauhan, A. Saxena e J. Murthy. «An Approach to Measure Security of Cloud Hosted Application». In: *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. Ott. 2013, pp. 1–6. DOI: 10.1109/CCEM.2013.6684427.

[34] Moon Cloud. *Moon Cloud Website*. 2017. URL: https://www.moon-cloud.eu/.

[35] United States. General Accounting Office. *Information Security: Agencies Need to Implement Consistent Processes in Authorizing Systems for Operation : Report to Congressional Requesters*. U.S. General Accounting Office, 2004. URL: https://books.google.it/books?id=jyTjnQAACAAJ.

[36] National Institute Of Standards. *NIST SP 800-53 r4 - Recommended Security Controls for Federal Information Systems*. 2013. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf.

[37] National Institute Of Standards. *NIST SP 800-60 v2r1 - Guide for mapping types of information and information systems to security categories*. 2003. URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication8060v2r1.pdf.

[38] National Institute Of Standards. *NIST SP 800-160 - Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. 2016. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf.

[39] National Institute Of Standards. *NIST SP 800-37 - Guide for Applying the Risk Management Framework to Federal Information Systems*. 2010. URL: http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf.

[40] National Institute Of Standards. *NIST SP 800-137 - Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations*. 2011. URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf.

[41] FedRAMP. *Guide to Understanding FedRAMP, v2*. 2014. URL: https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/482/2015/03/Guide-to-Understanding-FedRAMP-v2.0-4.docx.

[42] L. Taylor. «FedRAMP: History and Future Direction». In: *IEEE Cloud Computing* 1.3 (set. 2014), pp. 10–14. ISSN: 2325-6095. DOI: 10.1109/MCC.2014.54.

[43] FedRAMP. *FedRAMP Security Assessment Framework*. URL: https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/482/2015/01/FedRAMP-Security-Assessment-Framework-v2-1.pdf.

[44] FedRAMP. *FedRAMP Security Controls Preface*. URL: https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/482/2015/03/FedRAMP-Security-Controls-Preface-FINAL-1.pdf.

[45] FedRAMP. *FedRAMP HHH Low Baseline Security Controls*. URL: https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/482/2016/07/FedRAMP-Low-HHH-Baseline-Controls-2016-05-18.xlsx.

[46] FedRAMP. *FedRAMP HHH Moderate Baseline Security Controls*. URL: https://s3.amazonaws.com/sitesusa/wp-content/uploads/sites/482/2016/07/FedRAMP-Moderate-HHH-Baseline-Controls-2016-05-18.xlsx.