- Gli interventi manuali limitano la capacità di scalare e sfruttare al meglio gli strumenti e i controlli di sicurezza nativi del cloud. Considerare il software sviluppato in ottica di sicurezza e l'automazione come i migliori alleati in ambito cloud.

5.2.2.3 Best practice di secure design per le architetture serverless

Il serverless computing, noto anche come function as a service (FaaS), è una tendenza emergente nello sviluppo di applicazioni web che offre numerosi vantaggi rispetto all'architettura tradizionale basata su cloud: riduzione dei costi operativi, semplificazione dei processi, maggiore flessibilità, scalabilità e distribuzione più rapida. Tuttavia, per quanto vantaggioso, il serverless computing è comunque vulnerabile alle minacce alla sicurezza proprio come i servizi basati sul cloud, anche se in modi diversi. Da una parte, una architettura di questo tipo, riduce i rischi che coinvolgono i server, come ad esempio attacchi DDoS (distributed-denial-of-service). Elimina inoltre la necessità per gli sviluppatori di gestire le patch del sistema operativo. D'altro lato, il serverless computing solleva una nuova serie di problematiche di sicurezza che consentono agli hacker di eseguire azioni dannose sulle applicazioni. Tutte le funzioni serverless sono prive di stato, il che significa che i dati sensibili, comprese le sessioni utente, vengono trasmessi e memorizzati in una posizione esterna. Tale configurazione inevitabilmente aumenta il rischio di perdite di dati quando questi vengono spostati all'esterno del server.
Pertanto è opportuno disporre all'interno dell'organizzazione di controlli e pratiche di sicurezza da attuare in ambito serverless. Di seguito alcune pratiche utili ad incrementare il livello di sicurezza in tale ambito.

- Funzione di iniezione di dati relativi ad un evento
  - Le architetture serverless costituiscono una moltitudine di fonti di eventi, che possono innescare l'esecuzione di una funzione serverless. Tali funzioni possono elaborare input proveniente da qualsiasi tipologia di fonte di evento, e tale input può includere diversi formati di messaggio, a seconda del tipo di evento e della fonte di provenienza. Le parti costituenti di tali messaggi possono contenere dati controllati da possibili aggressori o comunque dannosi.
- Compromissione del processo di autenticazione
  - Le architetture Serverless promuovono una progettazione di sistema orientata ai microservizi e sono costituite da funzioni che vengono combinate e orchestrate per realizzare la logica di business complessiva del sistema. Alcune funzioni serverless possono esporre web API pubbliche, mentre altre possono consumare eventi provenienti da diversi tipi di fonti, come eventi di cloud storage, eventi di database NoSQL, segnali di telemetria generati dai dispositivi IoT o anche messaggi di notifica SMS. Pertanto è opportuno adottare schemi di autenticazione forti, capaci di fornire l'adeguato livello di controllo degli accessi e di protezione, a tutte le funzioni maggiormente rilevanti, i tipi di eventi e i trigger.
- Configurazione di deployment non sicura in ambito Serverless
  - I servizi cloud in generale, e le architetture serverless in particolare, per essere adattate ad ogni specifica esigenza, compito o ambiente circostante, offrono numerose personalizzazioni e impostazioni di configurazione. Alcune di queste impostazioni hanno implicazioni critiche sulla postura di sicurezza generale dell'applicazione e come tali, devono essere tenute in considerazione. Pertanto, non affidarsi alle impostazioni predefinite fornite dai fornitori di architetture serverless.
- Permessi di funzioni e ruoli eccessivamente privilegiati
  - Le applicazioni Serverless dovrebbero sempre seguire il principio del "minimo privilegio". Ciò significa che ad una funzione serverless dovrebbero essere concessi solo quei privilegi, che sono essenziali per eseguire la logica prevista. In un sistema in

cui tutte le funzioni condividono lo stesso set di permessi sovra-privilegiati, la presenza di una vulnerabilità in una singola funzione può degenerare in una catastrofe di sicurezza a livello di sistema.

- Monitoraggio e Logging di funzioni inadeguato
  - Rafforzare le impostazioni base o di default di logging per fornire un audit trail completo degli eventi di sicurezza. Tale configurazione dovrebbe includere elementi quali l'uso riuscito/fallito della chiave di accesso API, tentativi di invocazione di funzioni serverless con autorizzazioni inadeguate, distribuzione riuscita/fallita di nuove funzioni o configurazioni serverless, modifiche delle autorizzazioni di funzioni o dei ruoli di esecuzione, interazioni anomale o flussi irregolari tra funzioni serverless, connessioni in uscita avviate da funzioni serverless ed esecuzione di funzioni serverless o accesso ai dati da parte di un account esterno di terze parti non correlato al main account.
- Dipendenze non sicure da terze parti
  - Definire un processo per mantenere aggiornato un inventario riportante i pacchetti software e altre dipendenze con le relative versioni, scansionare il software alla ricerca di dipendenze vulnerabili note, rimuovere le dipendenze inutili, aggiornare le versioni dei pacchetti deprecate alle versioni più recenti e applicare tutte le patch software necessarie.
- Memorizzazione non sicura delle informazioni sensibili trattate dall'applicazione
  - Memorizzare tutti i dati sensibili trattati dalle applicazioni in un archivio crittografato sicuro e garantire che le chiavi crittografiche siano mantenute attraverso un'infrastruttura o servizio di gestione centralizzata. Tale tipologia di servizi sono normalmente disponibili nella maggior parte delle soluzioni architetturali serverless e cloud di mercato, che forniscono agli sviluppatori anche API sicure da poter utilizzare per integrarsi facilmente e perfettamente con gli ambienti serverless.
- Indisponibilità del servizio ed esaurimento delle risorse finanziarie
  - I vendors di soluzioni architetturali serverless definiscono i limiti predefiniti per l'esecuzione delle funzioni serverless. A seconda del tipo di limite e di attività, eventuali applicazioni mal progettate o configurate possono essere violate in modo tale da rendere la latenza nella risposta inaccettabile o addirittura per alcuni utenti inutilizzabile. Inoltre, un aggressore può spingere l'applicazione serverless per lunghi periodi di tempo, ad un "sovrautilizzo", causando essenzialmente un incremento della fatturazione mensile e infliggendo una significativa perdita finanziaria per l'organizzazione destinataria.
- Manipolazione o compromissione del flusso di esecuzione di una funzione Serverless
  - La possibilità di poter manipolare il flusso applicativo può aiutare eventuali aggressori a sovvertire la logica applicativa. Attraverso questa tecnica, un attaccante può a volte bypassare i controlli di accesso, elevare i privilegi dell'utente o persino sferrare un attacco di Denial of Service. In un sistema in cui esistono una moltitudine di funzioni e ciascuna di queste può invocarne un'altra, l'ordine di invocazione può essere critico nel raggiungimento della logica desiderata. Inoltre, la progettazione dell'applicazione potrebbe presumere che alcune funzioni siano invocate solo in scenari specifici e solo da invocazioni autorizzate. Assicurarsi che per ciascuna funzione vengano impostati gli adeguati permessi e controlli di accesso e, lì dove applicabile, impiegare una solida infrastruttura di gestione dello stato dell'applicazione.
- Gestione impropria degli errori e eccessiva verbosità dei messaggi di errore
  - Le possibilità di eseguire il debug linea per linea delle applicazioni serverless sono piuttosto limitate e di maggiore complessità rispetto alle classiche opzioni di debug