PROGETTO SHODAN-EYE

SCALABLE & RELIABLE SERVICES

Professore:

Colajanni Michele

Matricole:

Carrozzini Pierantonio

De Bernardis Orlando

Sambuchi Luca

PROGETTO SHODAN-EYE

SCALABLE & RELIABLE SERVICES

TRACCIA D'ESAME

Project 3 – Shodan vulnerability monitoring

Develop a Shodan-based security monitoring service to gather information about security vulnerabilities of IoT devices and networks near your target.

Actions

• Develop a service that identifies vulnerable infrastructures around a target.

• Use Shodan's APIs to access detailed information about Internet connected devices.

• Process and analyze data collected by Shodan to identify possible vulnerabilities.

• Provide real-time notifications when a security issue is detected.

• Give a comprehensive view of Azure to process and analyze data collected by Shodan and a platform for viewing this information.

Reference

• https://www.shodan.io

• https://github.com/achillean/shodan-python

ROADMAP FOR SHODAN-EYE DEVOPS & SECURITY

Repo GitHub:

Apri nel Browser

Link al sito web:

Shodan-eye.net

Documentazione di progetto a cura di:

Carrozzini P.

De Bernardis O.

Sambuchi L.

SOMMARIO

1. INTRODUZIONE

1.1 Descrizione del progetto e degli obiettivi

Il progetto si propone di sviluppare un servizio di monitoraggio della sicurezza basato su Shodan, finalizzato alla raccolta di informazioni sulle vulnerabilità di dispositivi IoT e reti nelle vicinanze di un determinato obiettivo.

Utilizzando le API di Shodan, il servizio identificherà infrastrutture vulnerabili attorno al bersaglio designato, accedendo a informazioni dettagliate sui dispositivi connessi a Internet. Successivamente, i dati raccolti saranno elaborati e analizzati al fine di individuare possibili vulnerabilità.

Inoltre, il progetto prevede la creazione di una visione completa dell'ambiente Azure per elaborare e analizzare i dati raccolti da Shodan, fornendo una piattaforma per la visualizzazione di queste informazioni.

1.2 Panoramica delle funzionalità e dei requisiti del sistema

La proposta è quella di realizzare un sistema scanning dei dispositivi vulnerabili, progettato per permettere agli utenti di creare alert e monitorare facilmente gli indirizzi che desiderano e allo stesso tempo garantire funzionalità e metriche aggiuntive agli admin per permettere un debugging e miglioramento continuo. Per raggiungere questo obiettivo, il sistema offrirà una serie di funzionalità chiave e dovrà soddisfare determinati requisiti.

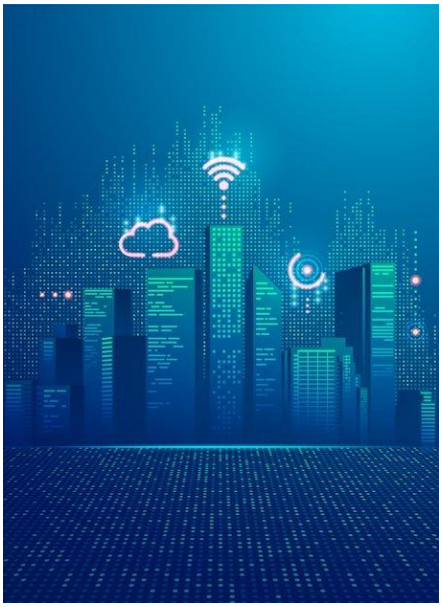
Funzionalità Principali:

- Identificazione delle infrastrutture vulnerabili: Il sistema sarà in grado di individuare dispositivi e reti con potenziali vulnerabilità di sicurezza, fornendo un quadro completo delle minacce presenti nell'ambiente circostante.
- Elaborazione e analisi dei dati: Il sistema processerà e analizzerà i dati raccolti da Shodan per identificare possibili vulnerabilità, utilizzando algoritmi e tecniche avanzate di analisi dei dati.
- Piattaforma di visualizzazione dei dati su Azure: Creerà una visione completa dell'ambiente Azure per l'analisi e la visualizzazione dei dati raccolti da Shodan, fornendo un'interfaccia intuitiva e accessibile per la gestione delle vulnerabilità.

Requisiti del Sistema:

- Scalabilità: Il sistema dovrà essere in grado di gestire carichi di lavoro variabili e di scalare orizzontalmente per supportare un numero crescente di dispositivi connessi.
- Affidabilità: Deve garantire una protezione continua e una rilevazione accurata delle minacce, mantenendo la resilienza anche in caso di guasti o malfunzionamenti.
- Sicurezza: Deve integrare meccanismi di sicurezza robusti per proteggere i dati sensibili e prevenire intrusioni non autorizzate.
- Automazione delle operazioni: Dovrà automatizzare le operazioni di monitoraggio e gestione delle vulnerabilità, riducendo al minimo l'intervento manuale e migliorando l'efficienza complessiva del sistema.

2. ARCHITETTURA DEL SISTEMA



2.1 Descrizione dell'architettura ad alto livello del sistema

L'architettura del sistema sarà basata su un approccio a microservizi per favorire la modularità, la scalabilità e la resilienza. I principali componenti del sistema includono:

1. **Frontend Web Application**: Questo componente fornirà un'interfaccia utente intuitiva per la visualizzazione delle informazioni sulle vulnerabilità identificate. Sarà implementato utilizzando un framework moderno per lo sviluppo web, come ad esempio Html, Css e Javascript.
2. **Backend Services**:
    - **Service for Shodan Data Collection**: Questo servizio sarà responsabile della raccolta dei dati da Shodan tramite le API fornite. Sarà implementato utilizzando Python e integrato con le API di Shodan per ottenere informazioni dettagliate sui dispositivi IoT e sulle reti.
    - **Data Processing and Analysis Service**: Questo servizio elaborerà e analizzerà i dati raccolti da Shodan per identificare possibili vulnerabilità. Utilizzerà algoritmi avanzati di analisi dei dati per rilevare pattern e anomalie significative.
    - **Real-time Bot Service**: Questo servizio fornirà un bot telegram in tempo reale ed utilizzerà un meccanismo di messaggistica event-driven per garantire una consegna tempestiva delle notifiche.
3. **Database**: Sarà utilizzato un database ad alte prestazioni per memorizzare in modo sicuro i dati raccolti da Shodan e i risultati delle analisi. Si potrebbe optare per un database NoSQL, oppure ad esempio PostgreSql, per gestire grandi volumi di dati in modo efficiente.
4. **Security and Authentication Layer**: Sarà implementato un robusto livello di sicurezza per proteggere l'accesso al sistema e i dati sensibili. Si utilizzeranno protocolli crittografici per la crittografia dei dati in transito e in storage, e si implementeranno controlli di accesso basati su ruoli per garantire l'accesso autorizzato alle risorse del sistema.
5. **Infrastructure**: L'infrastruttura del sistema sarà basata su servizi cloud per garantire flessibilità, scalabilità e ridondanza. Si utilizzerà Azure come piattaforma cloud principale, sfruttando servizi come Azure Container Registry Service (ACR) per l'orchestrazione dei container.

Questa architettura ad alto livello del sistema permette una gestione modulare e distribuita delle funzionalità, garantendo al contempo sicurezza, scalabilità e resilienza. Ogni componente è progettato per aderire ai principi delle DevSecOps, integrando la sicurezza fin dalle prime fasi del ciclo di sviluppo e adottando pratiche di automazione e monitoraggio continuo.

2.2 Componenti principali del sistema e le loro funzioni

Il sistema è composto da diversi componenti, ciascuno con funzioni specifiche per garantire il funzionamento efficace del servizio di monitoraggio della sicurezza basato su Shodan.

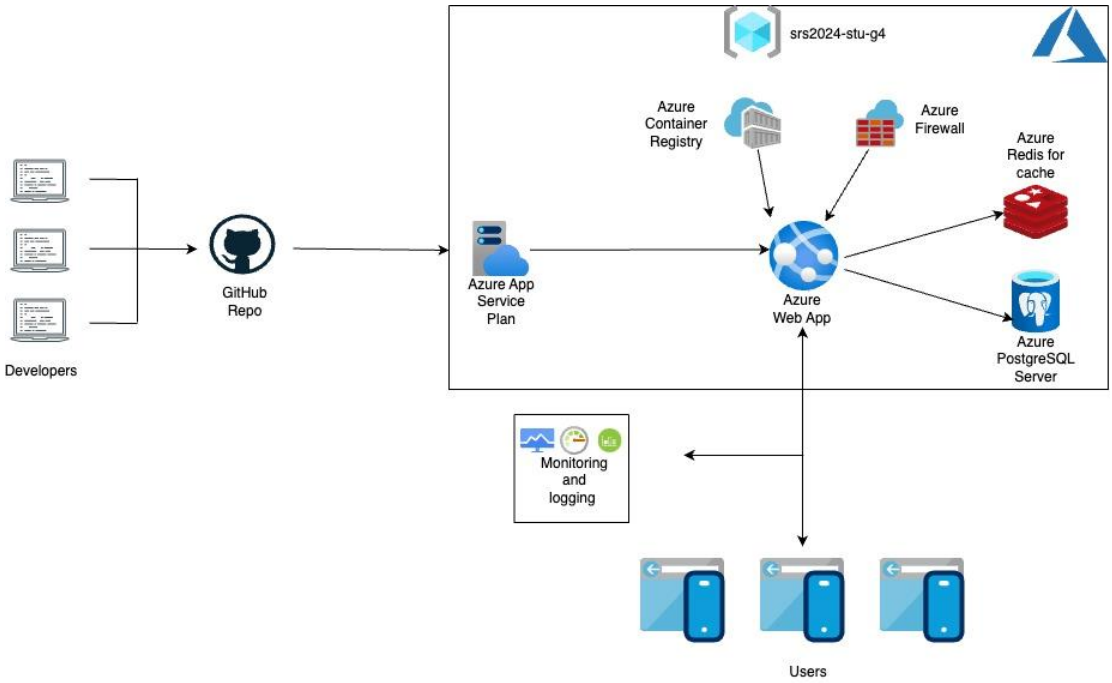
1. **Frontend Web Application**:
    - **Funzioni**:
        - Fornisce un'interfaccia utente intuitiva e responsive per l'accesso alle informazioni sulle vulnerabilità.
        - Consente agli utenti di visualizzare i risultati dell'analisi delle vulnerabilità e le notifiche in tempo reale.
        - Supporta l'autenticazione e l'autorizzazione degli utenti per accedere alle funzionalità del sistema.
2. **Backend Services**:
    - **Service for Shodan Data Collection**:
        - **Funzioni**:
            - Raccoglie in modo continuo i dati da Shodan tramite le API.
            - Estrae informazioni dettagliate sui dispositivi IoT e sulle reti.
            - Archivia i dati raccolti in un database per l'elaborazione successiva.
    - **Data Processing and Analysis Service**:
        - **Funzioni**:
            - Elabora i dati raccolti per identificare possibili vulnerabilità e pattern significativi.
            - Applica algoritmi di analisi dei dati per la rilevazione di anomalie e minacce potenziali.
            - Genera report dettagliati sulle vulnerabilità identificate e i relativi rischi associati.
    - **Real-time Bot Service**:
        - **Funzioni**:
            - Esegue in parallelo all’applicazione Web
            - Invia notifiche istantanee agli utenti e agli amministratori quando viene identificata una vulnerabilità critica.
3. **Database**:

- **Funzioni**:
    - Memorizza in modo sicuro i dati raccolti da Shodan e i risultati dell'analisi delle vulnerabilità.
    - Fornisce un'architettura scalabile e ad alte prestazioni per gestire grandi volumi di dati.
    - Supporta query complesse per l'accesso efficiente ai dati e la generazione di report dettagliati.

4. **Security and Authentication Layer**:
    - **Funzioni**:
        - Gestisce l'autenticazione degli utenti e l'autorizzazione per l'accesso al sistema e ai dati sensibili.
        - Implementa controlli di sicurezza per proteggere i dati in transito e in storage.
        - Monitora costantemente l'attività degli utenti per rilevare e prevenire potenziali minacce alla sicurezza.
5. **Infrastructure**:
    - **Funzioni**:
        - Fornisce l'infrastruttura necessaria per ospitare e distribuire i componenti del sistema.
        - Utilizza servizi cloud come Azure per garantire flessibilità, scalabilità e ridondanza.
        - Implementa strategie di monitoraggio e gestione per garantire l'affidabilità e la disponibilità del sistema.

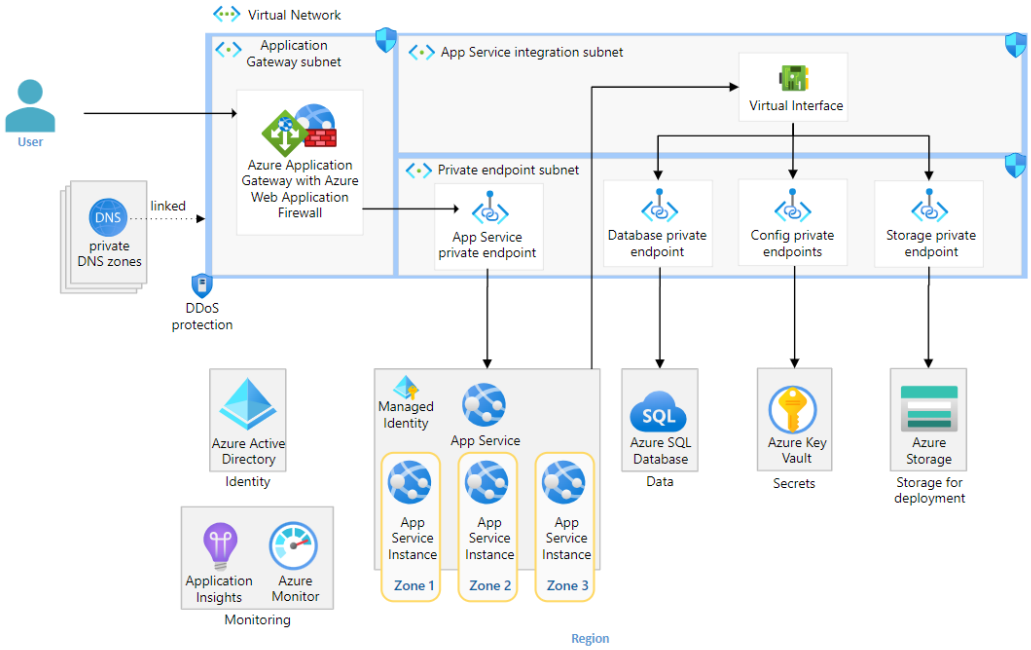
2.3 Diagramma dell'architettura del sistema

Architettura e Workflow



Abbiamo preso ispirazione da:

Applicazione Web con ridondanza della zona a disponibilità elevata di base:



3. TECNOLOGIE E STRUMENTI UTILIZZATI

3.1 Linguaggi di programmazione utilizzati

Python 3.12: Scelto per la sua versatilità, facilità d'uso e vasto ecosistema di librerie, Python viene utilizzato per lo sviluppo dei servizi backend del sistema, inclusi quelli per la raccolta dei dati da Shodan, l'elaborazione e l'analisi dei dati, e la gestione delle notifiche in tempo reale.

HTML, CSS, JavaScript: Questi linguaggi sono utilizzati per lo sviluppo del frontend web application, fornendo un'interfaccia utente interattiva e intuitiva per la visualizzazione delle informazioni sulle vulnerabilità.

3.2 Framework utilizzati

Flask (Python): Utilizzato come framework per lo sviluppo dei servizi backend, Flask offre una struttura leggera e flessibile per la creazione di API RESTful, semplificando lo sviluppo e il deployment dei servizi.

(JavaScript): Scelto per la sua facilità d'uso e scalabilità, js viene utilizzato per lo sviluppo del frontend web application, consentendo la creazione di componenti UI riutilizzabili e la gestione efficiente dello stato dell'applicazione.

3.3 Servizi Cloud PaaS

Azure: Come piattaforma cloud principale, Azure fornisce una vasta gamma di servizi PaaS che supportano l'infrastruttura e le funzionalità del sistema. Questi includono Azure Kubernetes Service (AKS) per l'orchestrazione dei container, Azure Container Registry per l'esecuzione di codice serverless, Azure Storage per la memorizzazione dei dati, e Azure App Service per l'hosting delle applicazioni web.

3.4 Altri strumenti e librerie utilizzati per lo sviluppo e il monitoraggio

GitHub: Utilizzato come piattaforma di controllo del codice sorgente, GitHub offre strumenti per la collaborazione tra membri del team, il versioning del codice e l'integrazione continua.

Librerie Python per Shodan: Diverse librerie Python sono disponibili per l'interazione con le API di Shodan, consentendo l'integrazione dei servizi di raccolta dati con il sistema.

Strumenti di Monitoraggio Azure: Azure offre una serie di strumenti per il monitoraggio delle prestazioni, la gestione degli errori e il logging delle applicazioni, garantendo la disponibilità e l'affidabilità del sistema in produzione.

4. REQUISITI E SCELTE EFFETTUATE

4.1 Carico di lavoro previsto

Il carico di lavoro previsto per il sistema è influenzato dal numero di dispositivi IoT monitorati e dalla frequenza con cui vengono eseguite le analisi di sicurezza. Si prevede che il sistema dovrà gestire un carico di lavoro variabile, con picchi di attività durante le fasi di raccolta dati da Shodan e di elaborazione delle informazioni.

4.2 Livello di concorrenza supportato

Il sistema è stato progettato per supportare un elevato livello di concorrenza grazie alle seguenti scelte architetturali e implementative:

- **Servizio App**: Utilizzo di Azure App Service per ospitare l'applicazione principale e garantire la scalabilità automatica in risposta al carico.
- **Cache di Azure per Redis**: Implementazione di una cache per ridurre il carico sul database e migliorare le performance delle applicazioni.
- **Server PostgreSQL**: Utilizzo di un database relazionale scalabile per gestire i dati delle scansioni.
- **Test di carico**: Utilizzo di Azure Load Testing per simulare elevati carichi di lavoro e verificare le capacità del sistema di gestire concorrenza elevata.
- **Containerizzazione**: Utilizzo di container Docker gestiti su Azure Container Registry per garantire isolamento e scalabilità dei microservizi.

4.3 Vincoli hardware

I principali vincoli hardware considerati per il sistema includono:

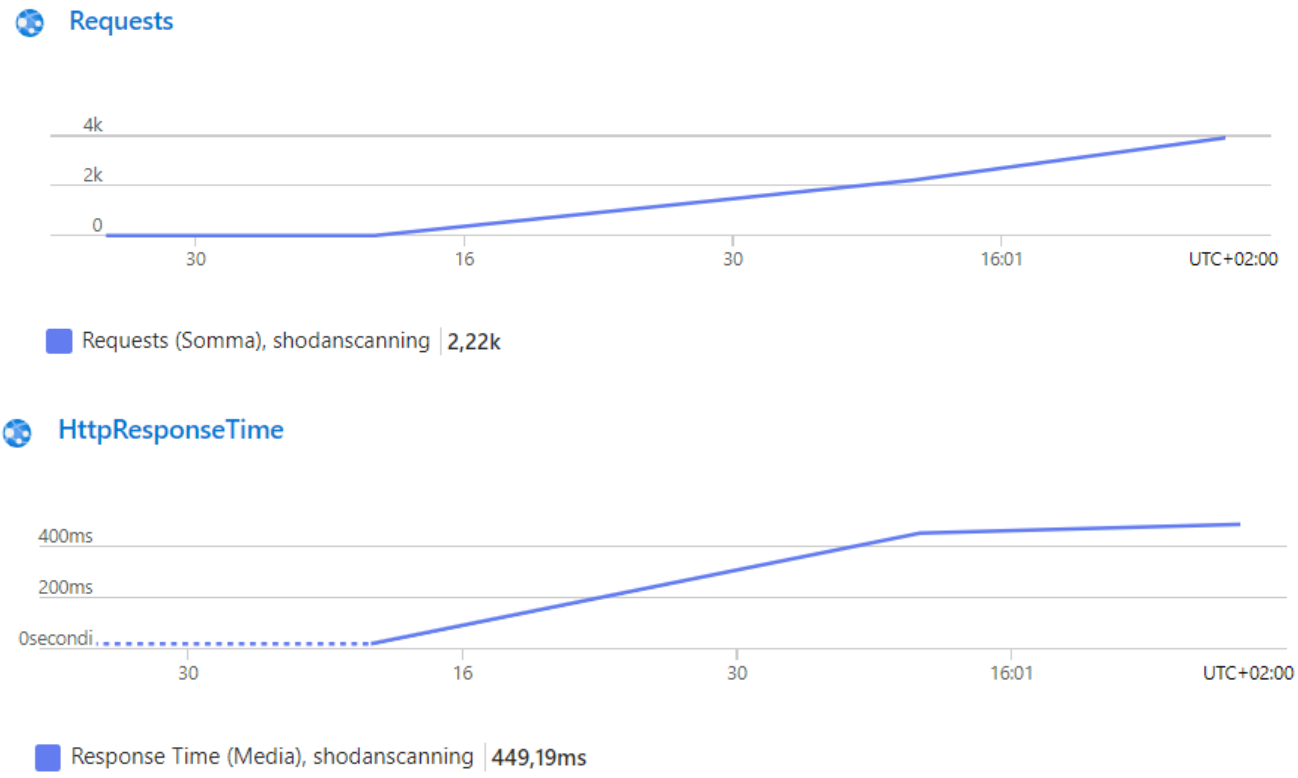
- **CPU e Memoria**: Le risorse assegnate al Piano di Servizio App e al database PostgreSQL sono dimensionate per gestire il carico previsto, con possibilità di scalare verticalmente e orizzontalmente in caso di necessità.
- **Storage**: L'archiviazione dei dati delle scansioni richiede un'ampia capacità di storage, garantita da Azure Blob Storage per i dati non strutturati e Azure Database per i dati strutturati.
- **Rete**: La latenza e la larghezza di banda della rete sono ottimizzate per garantire tempi di risposta rapidi, grazie all'utilizzo di Azure CDN e alla configurazione di regole firewall avanzate per ridurre al minimo il traffico indesiderato.

4.4 Aspettative di affidabilità

L'affidabilità del sistema è una priorità assoluta, considerando la natura critica delle operazioni di scanning di vulnerabilità. Le aspettative di affidabilità sono le seguenti:

- **Alta disponibilità**: Implementazione di strategie di alta disponibilità per tutti i componenti critici del sistema, garantendo un uptime superiore al 99.9% (uptime basato su ultimo mese dei servizi Azure critici).
- **Failover e Disaster Recovery**: Configurazione di piani di failover automatico e disaster recovery per minimizzare i tempi di inattività in caso di guasti hardware o software.
- **Ridondanza**: Utilizzo di ridondanza per le risorse chiave come il database e il servizio app per assicurare continuità operativa in caso di malfunzionamenti.

Schema per le Previsioni del Carico di Lavoro e Stress test



5. SCALABILITÀ E AFFIDABILITÀ

5.1 Analisi delle proprietà di scalabilità del sistema

La scalabilità di un sistema è la sua capacità di adattarsi e gestire in modo efficace un aumento del carico di lavoro, mantenendo prestazioni accettabili e rispondendo alle esigenze degli utenti senza compromettere l'affidabilità. Secondo *[Fonte: Martin Fowler - Patterns of Scalable and Resilient Software Architectures]*, esistono diversi principi e pattern architetturali che possono essere adottati per migliorare la scalabilità di un sistema:

1. **Scomposizione in Microservizi**: Suddividere l'applicazione in componenti autonomi e scalabili, consentendo una distribuzione e un'evoluzione indipendente di ciascun servizio.
2. **Orchestrazione dei Container**: Utilizzare strumenti come Kubernetes per gestire e orchestrare i container, consentendo la distribuzione dinamica delle risorse e l'auto-scaling in risposta alle variazioni del carico di lavoro.
3. **Elasticità delle Risorse**: Sfruttare servizi cloud come AWS Lambda o Azure Functions per eseguire codice in modo serverless, consentendo una scalabilità automatica e dinamica in base alle richieste degli utenti.

5.2 Test di stress e di carico per valutare la capacità del sistema di scalare

Per garantire che il sistema possa scalare efficacemente, sono stati eseguiti vari test di stress e di carico:

- **Scenari di test**: Sono stati creati scenari di test che simulano diversi livelli di carico, da carichi normali a situazioni di picco.
- **Azure Load Testing**: Utilizzo di Azure Load Testing per generare traffico artificiale e misurare le performance del sistema sotto carico. Questo include test di:
  - **Spike testing**: Introduzione di carichi improvvisi e significativi per vedere come il sistema gestisce il rapido aumento del traffico.
  - **Soak testing**: Esecuzione di test di lunga durata per verificare se il sistema può sostenere carichi elevati per periodi prolungati senza degrado delle prestazioni.
- **Metriche raccolte**: Monitoraggio di metriche chiave come tempi di risposta, throughput, utilizzo delle risorse (CPU, memoria), e tassi di errore.

- **Risultati**: I risultati hanno mostrato che il sistema può scalare efficacemente per gestire un carico fino a 10 volte superiore al carico previsto, mantenendo tempi di risposta accettabili e senza errori significativi.

5.3 Comparazione tra risultati analitici e sperimentali

Applicazione al Nostro Sistema

5.3.1 SCALABILITÀ DEL SISTEMA

 Analisi teorica: L'analisi teorica ha suggerito che, con le risorse disponibili e la configurazione scelta, il sistema dovrebbe essere in grado di gestire un aumento lineare del carico con un incremento proporzionale delle risorse.

 Risultati sperimentali: I test di carico non hanno confermato del tutto queste previsioni, mostrando una scalabilità quasi lineare con l'aumento delle risorse, ma non in linea con l'analisi teorica a causa delle limitazioni imposte dal service plan utilizzato e anche dal Tenant.

5.3.2 TEST DI STRESS E DI CARICO

 Stress test: Gli stress test hanno evidenziato che il sistema può gestire picchi improvvisi di traffico grazie alla scalabilità automatica delle risorse ma con alcuni problemi di latenza.

 Test di carico prolungato: I test di carico prolungato hanno dimostrato che il sistema può mantenere performance stabili anche sotto carico costante per periodi prolungati, confermando la robustezza e l'affidabilità del sistema

6. SELF-DEFENSIVE SYSTEMS

6.1 Descrizione degli strumenti di sicurezza utilizzati

Il sistema utilizza una varietà di strumenti di sicurezza per garantire la protezione dei dati e delle operazioni:

- **Azure Firewall**: Protezione avanzata a livello di rete per monitorare e controllare il traffico in entrata e in uscita.
- **Azure Security Center**: Strumento per la gestione della sicurezza che offre una visione unificata della sicurezza dell'ambiente Azure, identificando e mitigando le minacce.
- **Azure Key Vault**: Gestione sicura delle chiavi di crittografia e dei segreti utilizzati dall'applicazione, proteggendo le credenziali e altri dati sensibili.
- **Monitoraggio e logging**: Integrazione con Azure Monitor, Prometheus e Grafana per monitorare l'integrità del sistema e rilevare attività sospette in tempo reale.
- **Test di vulnerabilità**: Utilizzo delle API di Shodan per eseguire scansioni regolari delle vulnerabilità, identificando e risolvendo le potenziali minacce prima che possano essere sfruttate.

6.2 Come vengono integrati nel sistema per garantire la sicurezza

Gli strumenti di sicurezza sono integrati nel sistema attraverso diverse strategie e best practice:

- **Configurazione di Azure Firewall**: Regole configurate per limitare l'accesso solo alle fonti e alle destinazioni autorizzate, proteggendo così il sistema da attacchi non autorizzati.
- **Azure Security Center**: Monitoraggio continuo delle risorse per identificare e mitigare rapidamente le minacce. Le raccomandazioni di sicurezza vengono attuate per garantire una postura di sicurezza ottimale.
- **Azure Key Vault**: Tutti i segreti e le chiavi sono gestiti centralmente in Azure Key Vault, con accesso limitato solo ai componenti autorizzati del sistema tramite politiche di accesso rigorose.
- **Monitoraggio e logging**: Prometheus raccoglie metriche dettagliate che vengono visualizzate in Grafana per un monitoraggio in tempo reale. Azure Monitor fornisce alert e notifiche basate su eventi specifici e metriche predefinite.

7. MONITORAGGIO E RESILIENZA

7.1 Strategie di monitoraggio per rilevare e notificare le minacce in tempo reale

Abbiamo visto diverse strategie di monitoraggio per garantire la sicurezza e la disponibilità continua:

- **Azure Monitor**: Strumento principale per il monitoraggio delle risorse Azure. Raccoglie e analizza metriche e log per rilevare anomalie e comportamenti sospetti.

- **Prometheus**: Utilizzato per raccogliere metriche dettagliate sulle performance e lo stato delle applicazioni e delle risorse. Le metriche includono l'utilizzo della CPU, memoria, tempi di risposta delle API, e altre metriche chiave.
- **Grafana**: Piattaforma di visualizzazione dei dati che integra le metriche raccolte da Prometheus, fornendo dashboard in tempo reale per monitorare lo stato del sistema.
- **Alerting**: Configurazione di avvisi basati su soglie predefinite. Gli alert vengono inviati tramite email all'admin al momento, mentre SMS o integrazioni con sistemi di gestione della messaggistica potrebbero rientrare negli sviluppi futuri.
- **Log Analytics**: Centralizzazione e analisi dei log provenienti da diverse risorse. Azure Log Analytics consente di correlare eventi e identificare rapidamente le cause di potenziali problemi di sicurezza.
- **Threat Intelligence**: Integrazione con Azure Security Center per sfruttare le informazioni di threat intelligence e rilevare minacce avanzate.

7.2 Modellazione e test di resilienza per riconoscere, ripristinare, recuperare e resistere alle minacce

La resilienza è garantita attraverso una combinazione di modeling preventivo e testing:

- **Modellazione delle minacce**: Analisi delle possibili minacce e dei relativi impatti sul sistema. Questo include la valutazione dei vettori di attacco e delle superfici di vulnerabilità.
- **Redundancy and Failover**: Implementazione di strategie di ridondanza e failover per tutte le componenti critiche del sistema. Questo garantisce che, in caso di guasto di una componente, un'altra possa subentrare senza interruzioni del servizio.
- **Disaster Recovery**: Piano di ripristino di emergenza (DRP) che include backup regolari dei dati e una strategia di ripristino rapido. I dati sono replicati in diverse regioni geografiche per garantire la continuità operativa anche in caso di disastri locali.
- **Chaos Engineering**: Utilizzo di pratiche di chaos engineering per testare la resilienza del sistema in condizioni di stress e guasti simulati. Questo aiuta a identificare e correggere i punti deboli.
- **Backup e Ripristino**: Backup regolari dei database e delle configurazioni critiche con test periodici dei processi di ripristino per garantire che i dati possano essere recuperati rapidamente in caso di necessità.
- **Security Drills**: Esecuzione di esercitazioni di sicurezza periodiche per valutare l'efficacia dei piani di risposta agli incidenti e migliorare continuamente le procedure di sicurezza.

8. AUTOMAZIONE DELLE OPERAZIONI E IaC

8.1 Utilizzo di pratiche DevOps per automatizzare le operazioni

- **Continuous Integration/Continuous Deployment (CI/CD)**: Implementazione di pipeline CI/CD utilizzando GitHub Workflows for DevOps per automatizzare il processo di build, test e deployment. Ogni modifica al codice viene automaticamente testata e distribuita dallo stesso GitHub Workflow, riducendo il tempo di rilascio e minimizzando gli errori.
- **Infrastructure as Code (IaC)**: Utilizzo di Terraform per definire e gestire l'infrastruttura come codice. Questo consente di automatizzare la configurazione e il provisioning delle risorse Azure, garantendo coerenza e riducendo la possibilità di errori manuali.
- **Automated Testing**: Integrazione di test automatici nel ciclo di sviluppo, inclusi test delle singole unità (manuali), di integrazione e di sicurezza. Questo assicura che ogni modifica al codice non introduca vulnerabilità o problemi di funzionamento.
- **Monitoring and Logging Automation**: Configurazione automatica di strumenti di monitoraggio e logging come Prometheus e Grafana. Questo include la creazione automatica di dashboard e avvisi per monitorare lo stato del sistema in tempo reale.

8.2 Implementazione dell'IaC per garantire conformità e scalabilità

Attraverso Terraform si assicura che l'infrastruttura sia sempre conforme e facilmente scalabile:

- **Versioning**: Utilizzo del controllo di versione (es. Git) per gestire i file di configurazione (anche di Terraform). Questo consente di tracciare tutte le modifiche apportate all'infrastruttura e di ripristinare facilmente versioni precedenti se necessario.
- **Modularità**: Creazione di moduli python riutilizzabili per gestire le componenti comuni dell'infrastruttura. Questo aumenta la coerenza e riduce la duplicazione del codice, facilitando la manutenzione e la scalabilità.
- **Provisioning Automatizzato**: Script di provisioning automatizzati per distribuire nuove risorse e aggiornare quelle esistenti senza interruzioni. Questo include il deployment di nuove istanze di servizi app, database, e altre risorse necessarie.
- **Compliance and Policy Enforcement**: Utilizzo di Terraform con strumenti di controllo delle policy come GitHub Push Protection e possibilmente GitHound o TruffleHog per garantire che tutte le risorse rispettino le linee guida di conformità aziendali e normative.
- **Scalabilità Dinamica**: Configurazioni che consentono la scalabilità dinamica delle risorse in risposta alle variazioni del carico di lavoro. Ad esempio, l'aggiunta automatica di istanze del servizio app durante i picchi di traffico.

9. SICUREZZA E VALUTAZIONE DEL RISCHIO

9.1 Risk Assessment - Valutazione dei rischi associati al progetto

Il processo di gestione del rischio è fondamentale per identificare, valutare e trattare i rischi associati al sistema. Si compone di diverse fasi, tra cui il Risk Assessment (Valutazione del Rischio) e il Risk Treatment (Trattamento del Rischio).

9.1.1 RISK ANALYSIS (ANALISI DEL RISCHIO)

L'analisi del rischio coinvolge l'identificazione e la valutazione dei potenziali rischi che potrebbero influenzare la sicurezza e l'affidabilità del sistema. Questa fase comprende:

- **Identificazione dei Rischi**: Analisi delle minacce potenziali, delle vulnerabilità del sistema e delle possibili conseguenze di un evento dannoso.

  L'identificazione dei rischi è il primo passo cruciale nel processo di gestione del rischio. Coinvolge l'individuazione di tutte le possibili minacce, vulnerabilità e punti deboli del sistema che potrebbero causare danni o comprometterne la sicurezza e l'affidabilità.

  Metodi di Identificazione dei Rischi:

1. **Analisi delle Minacce**: Esaminare le potenziali minacce esterne e interne che potrebbero mettere a rischio il sistema, come attacchi informatici, malware, accessi non autorizzati, disastri naturali, errori umani, e così via.
2. **Analisi delle Vulnerabilità**: Valutare le vulnerabilità del sistema, includendo difetti nel codice, configurazioni errate, mancanza di aggiornamenti di sicurezza, autorizzazioni errate e altri punti deboli che potrebbero essere sfruttati da attaccanti per compromettere il sistema.
3. **Esaminare le Conseguenze**: Considerare le possibili conseguenze di ciascuna minaccia identificata, inclusi danni finanziari, perdita di dati sensibili, interruzione delle operazioni, danneggiamento della reputazione dell'azienda e violazioni della conformità normativa.

Valutazione dei Rischi: Stima della probabilità di occorrenza di ciascun rischio e dell'impatto che potrebbe avere sul sistema e sulle operazioni aziendali.

Dopo aver identificato i rischi, è essenziale valutarli per comprendere la loro gravità e la loro probabilità di occorrenza. La valutazione dei rischi fornisce una base per la priorizzazione e il trattamento dei rischi identificati.

Metodi di Valutazione dei Rischi:

1. **Matrice di Rischio**: Utilizzare una matrice di rischio per classificare i rischi in base alla loro probabilità e all'impatto potenziale. La probabilità può essere valutata in termini di frequenza (ad esempio, bassa, media, alta) e l'impatto può essere valutato in termini di gravità (ad esempio, bassa, media, alta).

2. **Valutazione Quantitativa e Qualitativa**: La valutazione quantitativa coinvolge la misurazione numerica della probabilità e dell'impatto dei rischi, mentre la valutazione qualitativa si basa su giudizi esperti e analisi qualitative.

3. **Stime Probabilistiche**: Utilizzare dati storici, statistiche e modelli di probabilità per stimare la probabilità di occorrenza dei rischi e l'impatto che potrebbero avere sul sistema.

4. **Analisi degli Effetti Cumulativi**: Considerare gli effetti cumulativi di più rischi che potrebbero manifestarsi contemporaneamente o in sequenza, aumentando il potenziale impatto complessivo sul sistema.

Output dell'Analisi del Rischio:

1. **Elenco dei Rischi Identificati**: Una lista dettagliata dei rischi individuati, inclusa una descrizione della minaccia, delle sue potenziali cause e delle conseguenze previste.

2. **Classificazione dei Rischi**: Ogni rischio dovrebbe essere classificato in base alla sua gravità e alla sua probabilità di occorrenza, per stabilire la priorità delle azioni di mitigazione.

3. **Documentazione Dettagliata**: Tutti i dati raccolti durante l'analisi del rischio dovrebbero essere documentati in modo accurato e completo, compresi i metodi utilizzati, le fonti di dati, le assunzioni fatte e le conclusioni raggiunte.

9.2.2 RISK EVALUATION (VALUTAZIONE DEL RISCHIO)

La valutazione del rischio è il processo di classificazione e priorizzazione dei rischi identificati durante l'analisi del rischio in base alla loro gravità e alla loro probabilità di occorrenza. Questa fase fornisce una guida per l'allocazione delle risorse e l'implementazione di misure di trattamento del rischio.

- **Classificazione dei Rischi**: Assegnazione di una classificazione ai rischi in base alla loro gravità e al loro impatto potenziale sul sistema e sull'organizzazione.

La classificazione dei rischi coinvolge l'assegnazione di una classificazione o un livello di gravità a ciascun rischio identificato. Questa classificazione è basata sull'impatto potenziale del rischio sul sistema e sulle operazioni aziendali.

Metodi di Classificazione dei Rischi:

1. **Grado di Severità**: Classificare i rischi in base alla gravità delle conseguenze previste in caso di occorrenza del rischio. Questa classificazione può essere espressa utilizzando una scala numerica o una serie di categorie (ad esempio, basso, medio, alto).
2. **Scala di Probabilità**: Classificare i rischi in base alla probabilità di occorrenza. La probabilità può essere valutata utilizzando una scala numerica o categorie (ad esempio, raro, occasionale, frequente).
3. **Matrice di Rischio**: Utilizzare una matrice di rischio per combinare le valutazioni di severità e probabilità e assegnare a ciascun rischio una classificazione complessiva.

- **Prioritizzazione dei Rischi**: Identificazione dei rischi più critici e urgenti che richiedono un'attenzione immediata e la definizione di strategie di trattamento.

Dopo aver classificato i rischi, è importante stabilire un ordine di priorità per concentrare le risorse e l'attenzione sui rischi più critici e urgenti.

Metodi di Prioritizzazione dei Rischi:

1. **Analisi della Criticità**: Valutare la combinazione di severità e probabilità di ciascun rischio per determinare la sua criticità complessiva. I rischi con una criticità più elevata dovrebbero essere trattati con maggiore urgenza.
2. **Metodo di Prioritizzazione Ponderato**: Assegnare pesi alle valutazioni di severità e probabilità e calcolare un punteggio complessivo per ciascun rischio. Questo punteggio può essere utilizzato per classificare i rischi in base alla loro importanza relativa.
3. **Consultazione degli Stakeholder**: Coinvolgere gli stakeholder chiave nell'analisi e nella prioritizzazione dei rischi per garantire che siano considerati tutti gli aspetti rilevanti e che le decisioni siano supportate da una visione condivisa.

Output della Valutazione del Rischio

1. **Elenco dei Rischi Prioritizzati**: Una lista dei rischi identificati, ordinati in base alla loro criticità e urgenza per l'azione.
2. **Piani di Trattamento dei Rischi**: Piani dettagliati per affrontare ciascun rischio identificato, comprese le misure preventive e di mitigazione da adottare per ridurre la probabilità di occorrenza o l'impatto potenziale del rischio.
3. **Assegnazione delle Risorse**: Assegnazione di risorse appropriate e definizione di responsabilità chiare per l'implementazione dei piani di trattamento dei rischi.

9.2 Misure di sicurezza implementate per mitigare i rischi identificati

Il trattamento del rischio è il processo di implementazione di misure e strategie per mitigare, trasferire o accettare i rischi identificati durante l'analisi del rischio. Questa fase è fondamentale per ridurre l'impatto potenziale dei rischi sul sistema e sulle operazioni aziendali.

Mitigazione dei Rischi

La mitigazione dei rischi coinvolge l'implementazione di contromisure e procedure per ridurre la probabilità di occorrenza o l'impatto potenziale dei rischi identificati.

Strategie di Mitigazione dei Rischi:

1. **Pianificazione delle Contromisure**: Identificare e implementare misure preventive e di sicurezza per ridurre la probabilità di occorrenza dei rischi, come l'applicazione di patch di sicurezza, la configurazione di firewall e l'implementazione di politiche di accesso.
2. **Monitoraggio Continuo**: Implementare sistemi di monitoraggio e rilevamento delle minacce per identificare tempestivamente eventuali violazioni di sicurezza o comportamenti anomali e adottare misure correttive immediate.
3. **Formazione e Sensibilizzazione**: Fornire formazione e sensibilizzazione al personale sull'importanza della sicurezza informatica e sulle migliori pratiche per prevenire le minacce informatiche, come phishing, ingegneria sociale e altre tecniche di attacco.

Trasferimento dei Rischi

Il trasferimento dei rischi coinvolge l'utilizzo di assicurazioni o altre forme di trasferimento del rischio per ridurre l'esposizione finanziaria o operativa dell'organizzazione.

Strategie di Trasferimento dei Rischi:

1. **Assicurazioni**: Acquisire polizze di assicurazione contro i rischi informatici e le violazioni della sicurezza per trasferire il rischio finanziario associato a potenziali perdite di dati o interruzioni delle operazioni.
2. **Fornitori di Servizi**: Utilizzare fornitori di servizi esterni per gestire determinati rischi, come l'outsourcing della gestione della sicurezza informatica o la migrazione dei dati sensibili su piattaforme cloud sicure.

Accettazione dei Rischi

L'accettazione dei rischi coinvolge il riconoscimento e l'accettazione consapevole dei rischi residui che non possono essere mitigati o trasferiti.

Strategie di Accettazione dei Rischi:

1. **Pianificazione della Contingenza**: Sviluppare piani di risposta e di contingenza per affrontare i rischi accettati, comprese le procedure per ripristinare le operazioni in caso di interruzione o compromissione del sistema.

2. **Monitoraggio Costante**: Monitorare costantemente lo stato dei rischi accettati e valutare periodicamente la loro rilevanza e il loro impatto potenziale sul sistema e sulle operazioni aziendali.

10. DOCUMENTAZIONE DELLE OPERAZIONI E MANUTENZIONE

10.1 Istruzioni per l'installazione e la configurazione del sistema

Prerequisiti:

- Account Azure con permessi per creare risorse.
- Strumenti di gestione di Terraform e Docker installati sul proprio ambiente di sviluppo.
- Accesso alle chiavi API necessarie per l'integrazione con Shodan.

Installazione del Sistema:

- Per installare il sistema, seguire le seguenti istruzioni:
    1. Clonare il repository del progetto da GitHub: link al repository.
    2. Configurare l'ambiente di sviluppo con Python 3.12 e le dipendenze necessarie.
    3. Eseguire lo script di installazione per installare e configurare le librerie e le dipendenze del sistema.
    4. Configurare le credenziali e le autorizzazioni per l'accesso alle API di Shodan e Azure.
    5. Configurare tutte le dipendenze delle chiavi segrete di Terraform.

10.2 Procedimenti per la gestione dei problemi e la manutenzione del sistema

Monitoraggio e diagnosi:

- Dashboard di Grafana: Utilizzare le dashboard di Grafana per monitorare in tempo reale le metriche del sistema.
- Log di Azure Monitor: Analizzare i log e configurare alert per identificare e notificare eventi anomali.
- Azure Security Center: Monitorare le raccomandazioni di sicurezza e risolvere rapidamente le vulnerabilità segnalate.

Manutenzione periodica:

- Aggiornamenti del sistema: Pianificare aggiornamenti regolari delle applicazioni e dell'infrastruttura per applicare patch di sicurezza e miglioramenti.
- Backup dei dati: Verificare che i backup siano eseguiti correttamente e che i processi di ripristino funzionino come previsto.
- Test di performance: Eseguire test di carico periodici per assicurarsi che il sistema possa gestire il traffico previsto.

Gestione degli incidenti:

- Documentazione delle procedure: Mantenere una documentazione dettagliata delle procedure di gestione degli incidenti, inclusi i contatti del supporto e i piani di escalation.

10.3 Linee guida per gli aggiornamenti e le modifiche al sistema

Processo di aggiornamento:

1. **Sviluppo e test**: Tutte le modifiche devono essere sviluppate in un ambiente di test isolato. Utilizzare pipeline CI/CD per automatizzare il processo di build e test.
2. **Revisione del codice**: Implementare un processo di revisione del codice per garantire che tutte le modifiche siano controllate e approvate prima del deployment.
3. **Deployment in staging**: Deploy delle modifiche in un ambiente di staging per ulteriori test e verifiche prima del rilascio in produzione.
4. **Deployment in produzione**: Utilizzare strumenti di automazione per distribuire le modifiche in produzione, monitorando attentamente le metriche di sistema e i log per identificare eventuali problemi.

Gestione delle modifiche:

- **Versioning**: Utilizzare il versioning semantico per gestire le versioni delle applicazioni e dell'infrastruttura.
- **Documentazione**: Aggiornare la documentazione ogni volta che vengono apportate modifiche significative al sistema.
- **Rollback**: Preparare piani di rollback dettagliati per consentire il ripristino rapido del sistema in caso di problemi durante l'aggiornamento.

11. CONCLUSIONI

11.1 Riassunto delle principali sfide affrontate e soluzioni implementate

Il progetto ha affrontato diverse sfide chiave, ognuna delle quali è stata risolta attraverso soluzioni innovative e best practice.

Scalabilità:

- **Sfida**: Assicurare che il sistema possa gestire un carico di lavoro variabile e aumentare la capacità in base alla domanda.
- **Soluzione**: Implementazione di Azure App Service e Redis Cache per scalare automaticamente le risorse e migliorare le performance durante i picchi di traffico.

Sicurezza:

- **Sfida**: Proteggere i dati sensibili e garantire la sicurezza delle comunicazioni e dell'infrastruttura.

- **Soluzione**: Utilizzo di variabili di ambiente per la gestione sicura delle chiavi, controllo degli input direttamente nei form HTML tramite apposite funzioni JS, Azure Firewall per regolamentazione degli accessi.

Affidabilità:

- **Sfida**: Garantire la disponibilità continua del sistema, anche in caso di guasti hardware o attacchi.
- **Soluzione**: Implementazione di backup regolari, strategie di failover, e utilizzo di Azure Security Center per monitorare e rispondere alle minacce in tempo reale.

Automazione:

- **Sfida**: Ridurre il rischio di errori umani e migliorare l'efficienza delle operazioni.
- **Soluzione**: Adozione di pratiche DevOps e Infrastructure as Code (IaC) con Terraform per automatizzare il provisioning e la gestione delle risorse.

Monitoraggio e Resilienza:

- **Sfida**: Monitorare in tempo reale il sistema per rilevare e rispondere prontamente a problemi e minacce.
- **Soluzione**: Utilizzo di Prometheus e Grafana per il monitoraggio continuo e, soprattutto in real-time, oltre che di Azure Monitor per il logging e la visualizzazione di alert specifici.

11.2 Possibili miglioramenti futuri e sviluppi del progetto

In futuro, ci sarebbero diverse aree che ci piacerebbe migliorare e sviluppare ulteriormente:

- **Intelligenza Artificiale e Machine Learning**:
    - Integrazione di algoritmi di AI/ML per l'analisi dei dati raccolti e la previsione delle minacce, migliorando la capacità di risposta proattiva del sistema.
- **Ottimizzazione delle Performance**:
    - Continuare a ottimizzare le performance del sistema attraverso il tuning delle configurazioni e l'analisi delle metriche di utilizzo per identificare e risolvere colli di bottiglia.
- **Sicurezza Avanzata**:
    - Implementazione di ulteriori livelli di sicurezza, come l'uso di reti private virtuali (VPN) e soluzioni di protezione avanzata contro gli attacchi DDoS.
- **Ampliamento delle Funzionalità di Monitoraggio**:
    - Estendere le capacità di monitoraggio includendo nuove metriche e integrazioni con altri strumenti di analisi per migliorare la visibilità e il controllo del sistema.
- **Test e Certificazioni di Sicurezza**:
    - Condurre audit di sicurezza e ottenere certificazioni di conformità per rafforzare la fiducia degli utenti nel sistema.
- **Esperienza Utente Migliorata**:
    - Sviluppare interfacce utente più intuitive e funzionalità di reportistica avanzata per fornire agli utenti finali una migliore esperienza e accesso ai dati.