Report Progetto: Analisi delle Dipendenze e Generazione di SBoM per Applicazioni React Native

Introduzione

Il report illustra il lavoro svolto per sviluppare uno script automatizzato in Node.js, volto a identificare le dipendenze di un'applicazione React Native, analizzarne le vulnerabilità e generare un Software Bill of Materials (SBoM) conforme al formato CycloneDX. Concentrandosi sulla crescente popolarità di React Native nel mobile development, lo studio analizza come lo SBoM può essere utilizzato per identificare e gestire le vulnerabilità di sicurezza nelle dipendenze software. Attraverso test su applicazioni reali, il progetto prevede l'individuazione di tecniche e strategie per estrapolare la lista delle dipendenze di un'app React Native, per poi automatizzarne l'esecuzione tramite creazione di opportuni script.

L'obiettivo principale è fornire uno strumento che permetta di valutare la sicurezza e la conformità delle applicazioni.

Obiettivi del Progetto

1. **Identificazione delle Dipendenze**: Rilevare automaticamente tutte le dipendenze (dirette e indirette) a partire dai file package.json e yarn.lock.
2. **Analisi delle Vulnerabilità**: Utilizzare i database di vulnerabilità, come OSV.dev e GitHub Advisory, per verificare se le dipendenze contengono vulnerabilità note.
3. **Generazione dello SBoM**: Creare un file JSON conforme a CycloneDX, contenente un elenco dettagliato dei componenti e delle relative informazioni di sicurezza.
4. **Automazione degli Aggiornamenti**: Implementare la possibilità di aggiornare automaticamente le dipendenze alla versione più recente e sicura.

Strumenti e Tecnologie Utilizzate

- **Linguaggi di Programmazione**: JavaScript/Node.js.
- **Strumenti e Librerie**:

    - npm e yarn per la gestione delle dipendenze.
    - axios per le richieste HTTP.
    - API di OSV.dev e GitHub Advisory Database per identificare vulnerabilità note.
    - Generazione dello SBoM in formato CycloneDX JSON.
- **Ambiente di Sviluppo**: Node.js v20.18.0 su Linux Kubuntu 24.04.

Workflow e Fasi dell'Attività

1. **Setup e Preparazione**:
    a. Creazione della struttura iniziale dello script.
    b. Configurazione di un token GitHub per accedere alle API.
    c. Creazione di directory per il salvataggio dei risultati, inclusi log di output e report.
2. **Identificazione delle Dipendenze**:
    a. Lettura dei file package.json e yarn.lock per ottenere l'elenco delle dipendenze.
    b. Filtraggio di dipendenze ridondanti o non valide.
    c. Aggiornamento delle dipendenze tramite confronto con le versioni più recenti disponibili su npm.
3. **Analisi delle Vulnerabilità**:
    a. Richiesta alle API di OSV.dev e GitHub Advisory.
    b. Combinazione dei risultati ottenuti da più fonti.
    c. Calcolo delle vulnerabilità uniche, con riepilogo per gravità (CRITICAL, HIGH, MODERATE, LOW).
4. **Generazione del Report**:
    a. Creazione di un file Markdown (vulnerability-report.md) che documenta:
        i. Pacchetti vulnerabili, con ID delle vulnerabilità e versioni fisse consigliate.
        ii. Riepilogo delle vulnerabilità suddiviso per gravità.
    b. Salvataggio del log della console in un file dedicato.
5. **Generazione dello SBoM**:
    a. Creazione di un file JSON in formato CycloneDX che include:
        i. Dipendenze del progetto.
        ii. Licenze associate.
        iii. Vulnerabilità conosciute.

Output Generati

- **File di Report (vulnerability-report.md)**:
    - Include un riepilogo delle vulnerabilità rilevate, organizzate per gravità.
    - Fornisce dettagli sui pacchetti vulnerabili e le versioni sicure suggerite.
- **File SBoM (sbom.json)**:
    - Elenco strutturato delle dipendenze con licenze, URL del repository e dettagli di sicurezza.
    - Conforme al formato CycloneDX per una facile integrazione con strumenti di analisi della sicurezza.
- **File di Log (console-output.txt)**:
    - Contiene tutti i dettagli operativi, inclusi errori e messaggi di debug.

Miglioramenti Implementati

1. **Filtraggio e Ottimizzazione dei Messaggi di Console**:
    a. Ho separato i messaggi importanti (es. riepilogo, generazione di file) da quelli di debug, reindirizzando questi ultimi al file di log.
2. **Validazione delle Dipendenze**:
    a. Ho aggiunto controlli specifici per verificare la validità di ogni pacchetto e versione prima dell'analisi.
3. **Interazione Utente**:
    a. Ho implementato un prompt interattivo per chiedere all'utente se procedere con l'aggiornamento automatico delle dipendenze.
4. **Generazione di un Riepilogo Finale**:
    a. Lo script fornisce un riepilogo chiaro del numero di componenti analizzati e delle vulnerabilità rilevate.
5. **Struttura Scalabile**:
    a. Lo script è stato progettato per essere modulare, permettendo di aggiungere facilmente altre fonti di vulnerabilità o miglioramenti futuri.

Risultati

- **Dipendenze Totali Analizzate**: XX.
- **Numero di Componenti Vulnerabili Identificati**: YY.
- **Dettaglio per Gravità**:
    - CRITICAL: Z.
    - HIGH: W.
    - MODERATE: V.

    - LOW: U.
- **SBoM Generato**:
    - Percorso del file generato: /path/to/sbom.json.

Conclusioni

1. **Valore del Lavoro Svolto**:
    a. Rafforzamento della sicurezza dei progetti React Native.
    b. Automazione del processo di analisi delle vulnerabilità.
    c. Creazione di un flusso standardizzato per la generazione di SBoM.
2. **Possibili Miglioramenti**:
    a. Integrazione con pipeline CI/CD.
    b. Estensione dello script per supportare altri ecosistemi software (es. Python, Ruby).
    c. Integrazione di un sistema di notifica automatica per avvisi di nuove vulnerabilità.

Sviluppi Futuri

1. **Estensione delle Fonti di Vulnerabilità**:
    a. Integrazione con altri database di vulnerabilità come Snyk o NVD.
2. **Supporto per Altri Linguaggi/Framework**:
    a. Estendere lo script per supportare progetti basati su altri ecosistemi (es. Python, Java).
3. **UI Grafica**:
    a. Creazione di un'interfaccia grafica per facilitare l'uso dello script da parte di utenti non tecnici.
4. **Test Automatizzati**:
    a. Implementare test end-to-end per garantire la robustezza dello script in diversi scenari.

Con questo progetto, ho esplorato l'importanza e l'implementazione di un Software Bill of Materials (SBoM) per le applicazioni sviluppate con React Native, contribuendo a migliorare la sicurezza software e l'efficienza dei processi di sviluppo.