UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Sicurezza del Codice: Analisi delle vulnerabilità di sicurezza del codice generato da ChatGPT-4 rispetto alle risposte di Stack Overflow e all'evoluzione da ChatGPT-3.5

RELATORE

Prof. Fabio Palomba

Università degli Studi di Salerno

CANDIDATO

Costantino Paciello

Matricola: 0512113661

Anno Accademico 2023-2024

Questa tesi è stata realizzata nel

Ai miei genitori e a mio fratello, pilastri della mia vita.

Ai miei nonni, ovunque siate, sempre nel mio cuore.

Abstract

L'adozione crescente dei Large Language Models (LLM) nell'ingegneria del software solleva importanti interrogativi sulla sicurezza del codice generato. Nonostante l'interesse della ricerca sul tema, permangono limiti nello stato dell'arte, in particolare nella valutazione delle vulnerabilità del codice generato dagli LLM, come ChatGPT. Questa tesi analizza le vulnerabilità di sicurezza nel codice generato da ChatGPT-4, confrontandolo con le risposte fornite da Stack Overflow e con una precedente analisi su ChatGPT-3.5. Utilizzando dataset esistenti, sono state poste a ChatGPT-4 le stesse domande estratte da Stack Overflow, e il codice prodotto è stato valutato per numero e tipologia di vulnerabilità secondo il Common Weakness Enumeration (CWE), mediante strumenti di analisi statica come CodeQL. I risultati mostrano un miglioramento della sicurezza nel codice generato da ChatGPT-4, con una riduzione significativa sia del numero che della varietà di vulnerabilità CWE rispetto a GPT-3.5 e alle risposte di Stack Overflow. Questo lavoro contribuisce a colmare le lacune nello stato dell'arte, evidenziando un'evoluzione positiva nella capacità degli LLM di produrre codice più sicuro. Le analisi svolte forniscono nuovi spunti per sviluppatori e ricercatori, indicando progressi nella sicurezza dei modelli ma evidenziando anche l'importanza di una supervisione continua per garantire una piena affidabilità degli LLM in ambito di sicurezza del codice.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Il software è diventato una componente essenziale della nostra vita quotidiana, impiegato in ambiti che spaziano dall'istruzione alla sanità, dai trasporti alla finanza, fino alle interazioni sociali. Ogni giorno utilizziamo applicazioni e sistemi che automatizzano, semplificano e migliorano innumerevoli attività. A livello economico, il settore software non solo facilita l'innovazione, ma è anche una forza trainante dell'economia globale, contribuendo significativamente al PIL di molte nazioni. Nel Regno Unito, ad esempio, il settore software porta ogni anno oltre 170 miliardi di sterline, rendendo evidente il suo ruolo come motore di crescita economica [1, 2].

L'importanza dello sviluppo software si manifesta anche nel modo in cui supporta e migliora servizi critici, cioè tutti quei sistemi in cui un malfunzionamento o una compromissione può causare gravi danni, come perdite di vite umane, impatti ambientali, danni economici significativi o minacce alla sicurezza nazionale. Nell'ambito sanitario, il software gioca un ruolo fondamentale nella gestione dei dati dei pazienti, nella diagnosi e nel trattamento. Le cartelle cliniche elettroniche e le piattaforme di telemedicina, ad esempio, consentono ai medici di accedere in tempo reale a dati vitali e di offrire assistenza a distanza, migliorando sia l'efficacia delle

cure sia l'accessibilità ai servizi sanitari [3].

In campo educativo, gli strumenti software rendono possibile la didattica a distanza, permettendo a milioni di studenti di apprendere online. Questi esempi dimostrano come il software sia fondamentale per il funzionamento e l'evoluzione dei settori che più influenzano la qualità della nostra vita. Tuttavia, con l'aumento della diffusione e della complessità dei sistemi software, aumenta anche il rischio legato alle vulnerabilità e ai malfunzionamenti. In contesti critici, come quello sanitario, una falla nel software può avere conseguenze gravi. Un errore in una macchina per la risonanza magnetica, ad esempio, potrebbe portare a dosaggi errati di radiazioni, con potenziali danni per il paziente. Analogamente, vulnerabilità nei sistemi di gestione delle cartelle cliniche possono esporre i dati sensibili dei pazienti a violazioni della privacy, mettendo a rischio sia la sicurezza personale che l'affidabilità del sistema sanitario [1, 2].

La qualità e la sicurezza del software sono quindi essenziali per proteggere l'integrità dei servizi di cui dipendiamo ogni giorno. La presenza di vulnerabilità nei sistemi critici può portare a danni economici, perdite di dati e compromissioni di servizi essenziali, sottolineando l'importanza di dotare gli sviluppatori di risorse e strumenti che possano aiutarli a produrre codice sicuro e affidabile.

1.1.1 Vulnerabilità e sicurezza

Le vulnerabilità software rappresentano difetti o debolezze nel codice che possono essere sfruttate per compromettere la sicurezza di un sistema. Tra le vulnerabilità più comuni si trovano le SQL injections, un tipo di attacco in cui dati non fidati vengono immessi in un'applicazione e interpretati come comandi di codice. Questo permette a un attaccante di accedere a informazioni sensibili o di manipolare i dati all'interno di un database, esponendo aziende e utenti a rischi significativi [4, 5].

Per esempio, in un sistema vulnerabile, un utente malintenzionato potrebbe immettere del codice SQL malevolo in un campo di input (come la password) e indurre il sistema a interpretare la stringa come una condizione vera, accedendo ai dati senza fornire credenziali valide. Questo tipo di vulnerabilità può consentire agli

attaccanti di ottenere accesso non autorizzato, manipolare dati sensibili o addirittura compromettere l'intero database [6, 7].

La lista delle vulnerabilità più pericolose, come il CWE Top 25 di MITRE, evidenzia altre falle di sicurezza, tra cui problemi di controllo degli accessi e difetti nei meccanismi di autenticazione, che possono essere sfruttati per ottenere accesso non autorizzato a dati e funzionalità critiche. Questi attacchi compromettono l'integrità e la disponibilità dei sistemi, portando a perdite di dati, interruzioni di servizio e danni economici che minano la fiducia degli utenti nei confronti delle applicazioni digitali [8].

In risposta a questi rischi, molte organizzazioni adottano strategie di sicurezza avanzate per prevenire vulnerabilità e attacchi. Secondo Orient Software, è essenziale implementare una gestione delle vulnerabilità che includa aggiornamenti costanti del software e valutazioni regolari di sicurezza per identificare e correggere i difetti prima che possano essere sfruttati. Anche l'OWASP, con il suo SSDLC (Secure Software Development Lifecycle), sottolinea l'importanza di introdurre misure di sicurezza in ogni fase del ciclo di sviluppo, dalla progettazione al rilascio, per ridurre la probabilità che le vulnerabilità raggiungano l'ambiente di produzione [6, 7].

Questi approcci alla sicurezza software sono essenziali per affrontare le sfide poste dalle vulnerabilità, che non solo influenzano il singolo sistema, ma possono avere impatti significativi a livello aziendale e istituzionale.

1.1.2 Evoluzione degli strumenti di supporto allo sviluppo software

In termini di sviluppo software, gli sviluppatori hanno a disposizione diversi metodi per ottenere supporto nella generazione di codice di alta qualità. Tra questi, Stack Overflow è stato a lungo il punto di riferimento principale per gli sviluppatori in cerca di risposte a domande su vari argomenti di programmazione. Questa piattaforma ha funzionato come un forum in cui i professionisti del settore possono condividere problemi, soluzioni e migliori pratiche, permettendo ai programmatori di tutto il mondo di beneficiare di un vasto archivio di conoscenze collettive.

Tuttavia, il panorama tecnologico sta cambiando con l'avvento dell'IA generativa [9], come i modelli di linguaggio GPT. Questi strumenti avanzati offrono nuove

possibilità per la generazione di codice, utilizzando l'apprendimento automatico per comprendere e produrre codice in modo più dinamico e contestualizzato rispetto ai tradizionali database di conoscenza. La tecnologia dietro GPT permette agli sviluppatori di ottenere risposte e codice generato automaticamente, potenzialmente più personalizzate e adatte alle specifiche esigenze del momento, rispetto alle soluzioni più statiche e basate esclusivamente sull'input umano tipiche di piattaforme come Stack Overflow.

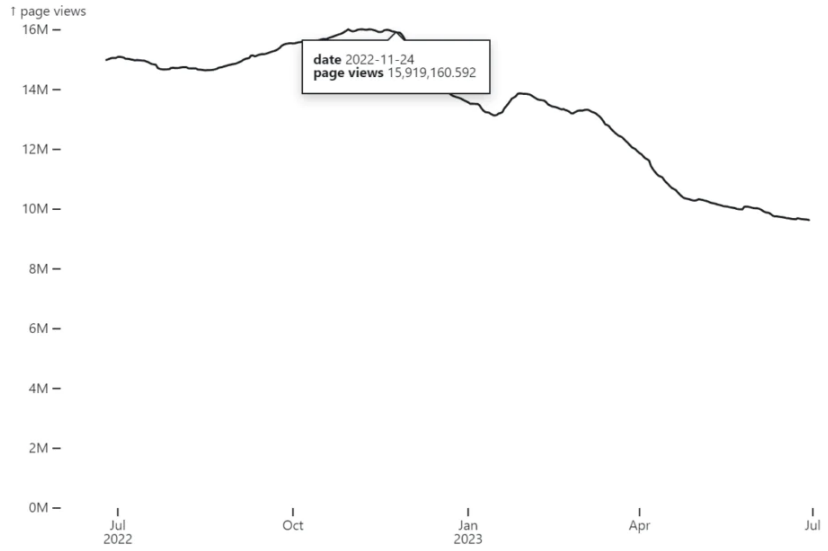

Figura 1.1: Visualizzazioni di pagine Stack Overflow.

L'industria dell'IA generativa sta sperimentando una crescita esponenziale, con previsioni che attestano un aumento del mercato da 36,06 miliardi di dollari nel 2024 a 356,1 miliardi entro il 2030, corrispondente a un tasso di crescita annuale del 46,47%. Nonostante un calo generale negli investimenti privati in IA, quelli destinati all'IA generativa hanno visto un incremento sostanziale, raggiungendo i 25,2 miliardi di dollari nel 2023, quasi nove volte l'investimento dell'anno precedente. Questo trend indica che, nel 2023, l'IA generativa ha costituito oltre un quarto degli investimenti complessivi nel settore dell'IA [10]. Questa espansione riflette non solo una crescente capacità delle applicazioni IA nel settore software, ma anche un aumento della fiducia nell'utilizzo di queste tecnologie per affrontare sfide di programmazione sempre più complesse.

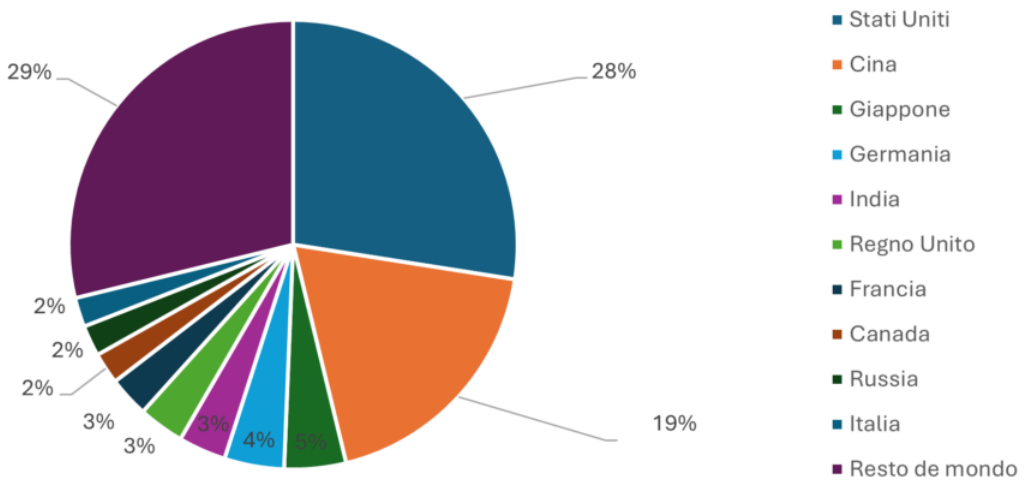

Figura 1.2: Mercato mondiale dell'intelligenza artificiale e confronto tra i principali paesi.

1.2 Motivazioni e obiettivi

Negli ultimi anni, i modelli di linguaggio di grandi dimensioni (LLM), come GPT-3.5 e GPT-4, hanno dimostrato una crescente capacità in vari ambiti di elaborazione del linguaggio naturale, tra cui la correzione grammaticale e la generazione di codice, guadagnando rapidamente popolarità e diffusione anche tra sviluppatori e professionisti della sicurezza informatica. Questi modelli sono stati utilizzati per migliorare processi come l'analisi del codice, l'identificazione delle vulnerabilità e persino la generazione di patch di correzione, attività che tradizionalmente richiedono competenze specialistiche di alto livello. Tuttavia, le ricerche esistenti indicano che, nonostante le potenzialità, gli LLM generici come ChatGPT mostrano ancora delle limitazioni rispetto a modelli ottimizzati per specifici compiti di sicurezza.

Un aspetto cruciale emerso dagli studi riguarda l'efficacia limitata degli LLM nell'affrontare problemi di sicurezza informatica senza una fase di addestramento specializzato. La mancanza di fine-tuning su dataset specifici del dominio, come quelli contenenti pattern di vulnerabilità ricorrenti o set di dati curati su falle di sicurezza, ostacola la capacità di questi modelli di rilevare e classificare con precisione le vulnerabilità. Infatti, strumenti di sicurezza progettati ad hoc, come AIBugHunter, utilizzano tecniche di fine-tuning su dataset dettagliati per affrontare attività di rilevamento e correzione delle vulnerabilità con una precisione e un'efficacia significativamente superiori, in particolare in compiti che richiedono una conoscen-

za approfondita di schemi di vulnerabilità e classificazioni standardizzate, come il Common Weakness Enumeration (CWE) [11, 12].

Queste limitazioni pongono interrogativi sul ruolo che gli LLM possono realmente avere nella scrittura di codice sicuro, evidenziando la necessità di un confronto approfondito tra le risposte fornite da modelli di linguaggio come GPT-3.5 e GPT-4 e quelle ottenute da fonti più tradizionali come StackOverflow. L'obiettivo di questa tesi è esplorare queste differenze, con l'intento di valutare se e in che misura gli LLM possano essere impiegati in modo affidabile per la generazione di codice sicuro.

Per affrontare queste sfide, questa tesi ha implementato un metodo di ricerca strutturato in diverse fasi. Inizialmente, è stata condotta una revisione sistematica della letteratura per identificare le vulnerabilità più comuni e i metodi di attacco prevalenti. Successivamente, sono stati generati snippet di codice utilizzando GPT-4 e applicato tecniche di analisi statica del codice attraverso CodeQL per rilevare e classificare le vulnerabilità. Queste sono state poi confrontate con le vulnerabilità identificate nei codici generati da GPT-3.5 e con le risposte di Stack Overflow, al fine di evidenziare le differenze tra le piattaforme e i miglioramenti tra le due versioni del modello.

Questo approccio ha permesso di valutare l'efficacia degli LLM nell'ambito della sicurezza informatica e di esplorare le differenze significative nella generazione di codice rispetto a fonti più tradizionali. L'obiettivo principale è stato determinare la fattibilità dell'uso degli LLM nella produzione di codice sicuro, mettendo in luce sia i progressi che le aree necessarie di miglioramento.

1.3 Risultati ottenuti

L'esame iniziale del comportamento di GPT-4 nel generare codice ha rivelato notevoli progressi rispetto a GPT-3.5, particolarmente nella diminuzione delle vulnerabilità identificate. L'analisi di 87 snippet di codice ha evidenziato che GPT-4 ha registrato solamente 101 vulnerabilità, un deciso calo rispetto alle 248 rilevate in GPT-3.5 e alle 302 osservate nei codici provenienti da StackOverflow. Queste cifre non solo riflettono un miglioramento nella qualità del codice generato da GPT-4, ma segnalano anche un avanzamento critico verso una produzione più sicura e affidabile.

Inoltre, la varietà di errori CWE riscontrati è stata sensibilmente inferiore, con GPT-4 che ha mostrato solo 10 tipi di CWE contro i 19 di GPT-3.5 e i 22 di StackOverflow. Questi risultati preliminari suggeriscono un'evoluzione significativa nell'efficacia con cui GPT-4 affronta le problematiche di sicurezza nel codice, promettendo potenziali impieghi più estesi e sicuri degli LLM nella programmazione.

1.4 Struttura della tesi

La presente tesi è articolata in cinque capitoli principali, ognuno dedicato a esplorare differenti aspetti del tema trattato.

- **Capitolo 1: Introduzione**, fornisce una panoramica generale dell'argomento, definendo gli obiettivi e la motivazione dietro lo studio delle vulnerabilità nei codici generati dai Large Language Models (LLM) e le risposte di Stack Overflow.
- **Capitolo 2: Background e stato dell'arte**, esplora i lavori precedenti e le teorie fondamentali relative alla sicurezza del software e all'impiego degli LLM nella generazione di codice, stabilendo il contesto teorico per la ricerca.
- **Capitolo 3: Metodo di ricerca**, dettaglia l'approccio metodologico adottato, comprese le tecniche di raccolta dati e gli strumenti di analisi utilizzati, come CodeQL e la comparazione con le metriche del Common Weakness Enumeration (CWE).
- **Capitolo 4: Analisi dei risultati**, presenta i dati raccolti e le analisi effettuate, discutendo come le vulnerabilità sono state identificate, confrontate e valutate tra le diverse fonti di codice.
- **Capitolo 5: Conclusioni**, riassume i risultati ottenuti, discute le implicazioni dei risultati e suggerisce possibili direzioni future per la ricerca in questo campo.

CAPITOLO 2

Background e stato dell'arte

Questo capitolo descrive i fondamenti degli LLM e le loro applicazioni nella generazione di codice, evidenziando gli studi precedenti relativi alla sicurezza del codice generato e le metodologie comunemente adottate per l'identificazione delle vulnerabilità.

2.1 Large Language Model

I Large Language Models (LLM) sono una tecnica avanzata di intelligenza artificiale specializzata nel riconoscere e generare linguaggio. Questi modelli sono definiti "grandi" per via dell'enorme quantità di dati su cui vengono addestrati. Gli LLM operano sulla base di reti neurali di tipo transformer e sono formati tramite vasti set di dati, spesso derivati da Internet, che possono raggiungere milioni di gigabyte di testo. Utilizzando tecniche di deep learning, questi modelli elaborano dati non strutturati, come testi, in modo probabilistico, cioè basandosi su calcoli di probabilità per fare previsioni o prendere decisioni. Attraverso questo processo, gli LLM riescono a comprendere e manipolare la struttura del linguaggio, ossia come sono organizzate le parole e le frasi. Questo permette al modello di distinguere autonomamente tra diversi tipi di contenuto (ad esempio, capire se un testo è una domanda o una di-

chiarazione) senza bisogno di intervento umano diretto o istruzioni esplicite, grazie all'apprendimento automatico. Infine, gli LLM sono raffinati attraverso processi di ottimizzazione per specializzarli in compiti specifici determinati dal programmatore, come interpretare domande, generare risposte coerenti, o tradurre testi tra diverse lingue. Questo affinamento mira a potenziare le loro funzionalità per rispondere precisamente alle esigenze delle applicazioni per cui sono destinati [13].

2.1.1 Storia e sviluppo degli LLM

La storia degli LLM inizia con le reti neurali negli anni '50 e '60, segnando i primi passi verso l'elaborazione del linguaggio naturale attraverso l'intelligenza artificiale. Tuttavia, è stato con l'introduzione della backpropagation negli anni '80 che le reti neurali hanno iniziato a essere impiegate in applicazioni pratiche complesse. Negli anni '90, modelli basati su catene di Markov e reti neurali ricorrenti (RNN) hanno aperto la strada ai primi modelli linguistici, sebbene con capacità limitate rispetto agli standard moderni.

L'avanzamento significativo nel campo dell'IA è avvenuto con l'avvento del deep learning all'inizio degli anni 2010, favorito dall'aumento della potenza di calcolo e dalla disponibilità di grandi dataset. Il vero cambiamento paradigmatico si è verificato con l'introduzione dei modelli Transformer nel 2017, che hanno migliorato l'elaborazione delle sequenze di dati e superato le limitazioni delle RNN e LSTM (Long Short-Term Memory), facilitando l'analisi parallela e migliorando le prestazioni nei vari compiti di NLP (Natural Language Processing).

Con la serie GPT, OpenAI ha sfruttato l'architettura Transformer per creare modelli che non solo comprendono meglio il linguaggio umano ma sono anche capaci di generare testo altamente coerente e contestualizzato. GPT-3, in particolare, con i suoi 175 miliardi di parametri, ha segnato una svolta, stimolando vasto interesse e una molteplicità di applicazioni pratiche nel campo [14].

2.1.2 Rivoluzione del Deep Learning e l'emergere dei Transformer

La rete neurale Transformer è un'architettura innovativa di reti neurali artificiali, introdotta nel 2017 da Vaswani et al. nell'articolo "Attention Is All You Need" [15]. Fin

dalla sua pubblicazione, è divenuta una delle architetture più influenti e ampiamente adottate nei campi dell'elaborazione del linguaggio naturale e dell'apprendimento automatico, grazie alla sua capacità di gestire efficacemente le dipendenze a lungo raggio e l'elaborazione parallela dei dati.

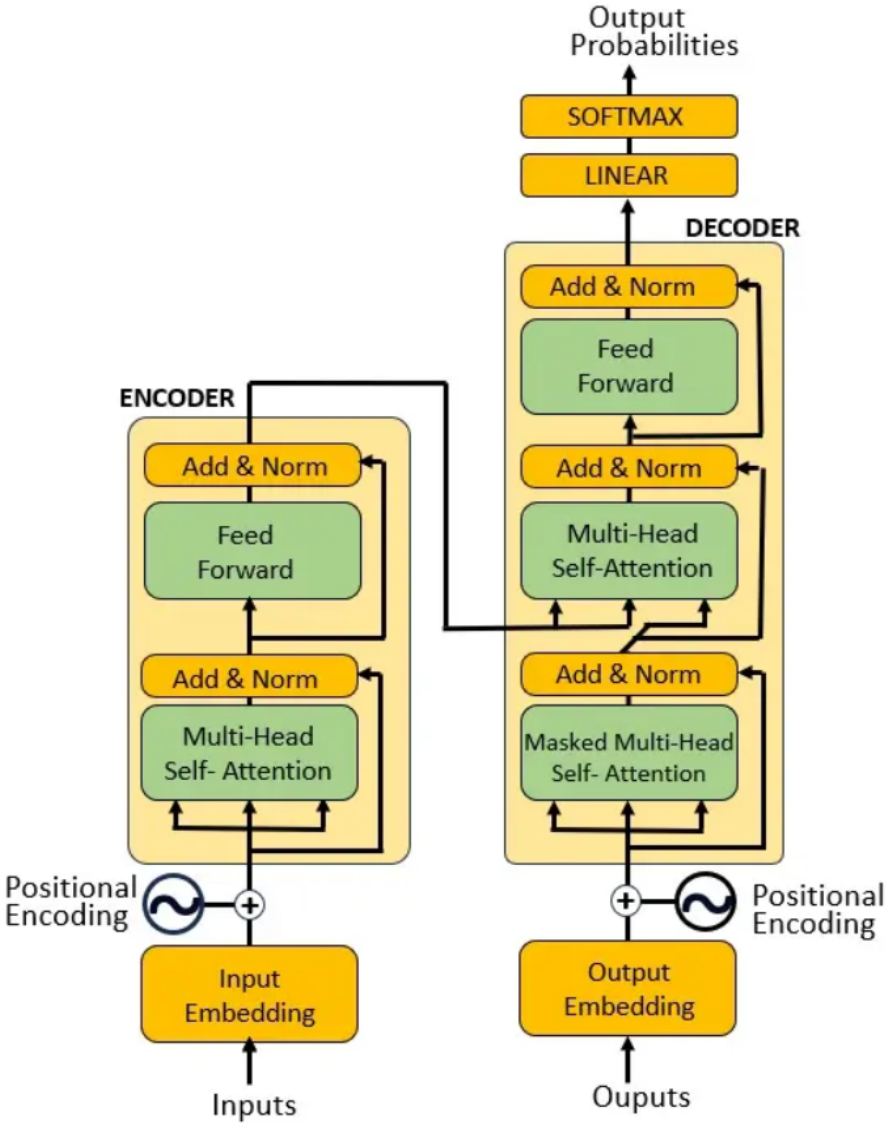

Figura 2.1: Architettura di una rete neurale Transformer.

L'architettura Transformer si basa su due elementi fondamentali:

1. **Meccanismo di auto-attenzione**: Questo meccanismo permette al modello di attribuire pesi variabili alle parole in una sequenza in funzione del loro contesto. Grazie a questo approccio, il modello può cogliere in modo efficace le relazioni tra parole anche distanti, gestendo così le dipendenze a lungo raggio nel testo.

In pratica, la rete è in grado di "concentrarsi" su specifiche parti della sequenza in modo dinamico e adattivo.

2. **Struttura senza ricorrenza**: A differenza delle reti neurali ricorrenti (RNN), che elaborano le sequenze in modo sequenziale, i Transformer operano in parallelo. Questa caratteristica consente di trattare l'input in modo altamente efficiente e scalabile, permettendo l'addestramento di modelli con un elevato numero di parametri e migliorando le prestazioni su compiti complessi.

L'architettura Transformer è composta da più strati, o blocchi, che possono essere impilati uno sopra l'altro. Ogni blocco include diversi sotto-moduli chiave:

- **Multi-Head Self-Attention**: Questo modulo calcola l'auto-attenzione su una sequenza di input, permettendo al modello di assegnare pesi variabili a ciascuna parola in base al contesto, migliorando così la capacità di cogliere relazioni complesse tra parole.
- **Feed Forward**: Strati di reti neurali feed-forward vengono utilizzati per elaborare le rappresentazioni generate dall'auto-attenzione, producendo rappresentazioni intermedie che arricchiscono il contenuto informativo del modello.
- **Add & Norm**: Questi strati servono a stabilizzare il processo di apprendimento, applicando normalizzazioni che mitigano il rischio di instabilità durante l'addestramento.
- **Connessioni residue**: Ogni blocco è dotato di connessioni residue, che facilitano la propagazione del gradiente e migliorano l'efficienza dell'addestramento.

Le reti Transformer possono essere configurate in vari modi, tra cui l'architettura encoder-decoder per la traduzione automatica, il solo encoder per il riconoscimento di entità e il solo decoder per la generazione di testo. Questa struttura versatile e adattabile è alla base di molti dei Large Language Models (LLM) di successo degli ultimi anni, come BERT, GPT-3 e altri [16].

Grazie a queste innovazioni, i Transformer hanno migliorato notevolmente le prestazioni in vari compiti di NLP, spingendo ulteriormente i confini di ciò che l'intelligenza artificiale può fare nel campo del linguaggio naturale.

2.1.3 Sfide attuali e prospettive future

Nonostante i progressi, lo sviluppo degli LLM presenta notevoli sfide, come l'enorme consumo di risorse computazionali e la generazione di output talvolta inaccurato o biasato, sollevando preoccupazioni etiche significative. Tali questioni evidenziano la necessità di indirizzare problemi di bias e di responsabilità nel design e nell'uso degli LLM.

Il futuro della ricerca sugli LLM si orienta verso il miglioramento dell'efficienza energetica, la riduzione del bias e l'ampliamento delle capacità di comprensione del linguaggio. L'obiettivo è integrare questi modelli avanzati in modo sempre più efficace e etico in settori critici come l'educazione, la salute e l'intrattenimento.

2.2 Large Language Models e generazione di codice

Tra le numerose applicazioni degli LLM, la produzione automatizzata di codice rappresenta un notevole vantaggio nel campo dello sviluppo software, rendendo la programmazione più accessibile a sviluppatori di ogni livello di esperienza. Attraverso l'inserimento di comandi testuali che descrivono le funzionalità desiderate, gli strumenti di sviluppo basati su AI generativa sono capaci di generare automaticamente il codice necessario. Queste tecnologie presentano inoltre il potenziale per modernizzare codici obsoleti e per tradurre codici tra diversi linguaggi di programmazione. Integrando l'AI nei toolkit per sviluppatori, queste soluzioni possono fornire suggerimenti di codice di elevata qualità basati sugli input degli utenti. I codici generati in autonomo migliorano la produttività degli sviluppatori, ottimizzando i loro flussi di lavoro con soluzioni rapide, gestendo le routine di codifica quotidiane. Questi strumenti possono altresì contribuire alla rilevazione di errori di programmazione e all'identificazione di potenziali vulnerabilità di sicurezza [17].

2.3 Sicurezza del codice generato

Recenti studi hanno evidenziato che il codice generato dagli LLM può contenere vulnerabilità di sicurezza, alcune delle quali piuttosto critiche. Complessivamente, i

risultati hanno mostrato che il codice prodotto da ChatGPT presentava 248 vulnerabilità, classificate secondo la Common Weakness Enumeration (CWE). Tra i principali aspetti problematici vi erano la gestione delle eccezioni, la sicurezza delle sessioni e l'uso delle funzioni di crittografia, mettendo in luce una serie di problemi critici che potrebbero compromettere la sicurezza delle applicazioni sviluppate. Questi risultati indicano che, sebbene ChatGPT possa essere uno strumento potente per la generazione rapida di codice, è fondamentale che gli sviluppatori non si affidino ciecamente al codice generato senza adeguati controlli [18].

2.3.1 Common Weakness Enumeration (CWE)

Il Common Weakness Enumeration (CWE) è un elenco completo di debolezze software e hardware, progettato per aiutare i professionisti della sicurezza nell'identificazione delle vulnerabilità all'interno dei propri sistemi. L'elenco è organizzato in categorie che riflettono il tipo di vulnerabilità, facilitando così la priorità degli interventi e lo sviluppo di strategie di mitigazione efficaci. Il CWE si rivela uno strumento essenziale per i professionisti della sicurezza, poiché permette loro di individuare e affrontare tempestivamente le vulnerabilità, prevenendo così eventuali sfruttamenti da parte di attori malevoli. Il CWE trova ampia applicazione nelle valutazioni delle vulnerabilità, supportando i professionisti nell'identificazione e nella classificazione delle debolezze presenti nei sistemi software. Grazie al CWE, è possibile rilevare punti deboli nelle fasi di progettazione, implementazione e configurazione del software, permettendo lo sviluppo di misure di mitigazione mirate. Inoltre, il CWE consente di dare priorità alle vulnerabilità in base alla loro gravità, ottimizzando così la distribuzione delle risorse per affrontare prima le minacce più critiche [19].

2.3.2 CodeQL

CodeQL è uno strumento avanzato di analisi statica del codice, progettato per identificare vulnerabilità e potenziali problemi all'interno di repository software. Sviluppato da GitHub, si distingue per l'uso di un linguaggio di query chiamato QL, che permette di trattare il codice come un database interrogabile. Questo ap-

proccio consente ai professionisti della sicurezza di eseguire analisi dettagliate, sia utilizzando query predefinite fornite da GitHub sia creando query personalizzate per rispondere a esigenze specifiche. Uno degli aspetti più potenti di CodeQL è la sua capacità di analizzare il codice sorgente e individuare vulnerabilità conosciute, come errori di sicurezza legati al Common Weakness Enumeration (CWE), contribuendo a migliorare la sicurezza del software. CodeQL è ampiamente utilizzato attraverso GitHub Actions, permettendo di automatizzare le scansioni di sicurezza e ricevere avvisi su eventuali vulnerabilità rilevate. Grazie alla sua flessibilità, può essere configurato per analizzare diversi linguaggi di programmazione e supporta l'integrazione di query specifiche per scenari particolari, migliorando la precisione dell'analisi [20].

2.4 Studi precedenti e metodi di analisi

Le crescenti preoccupazioni sulla sicurezza del codice generato dagli LLM hanno portato le comunità scientifiche e professionali a sviluppare tecniche più avanzate per l'analisi e la mitigazione dei rischi corrispondenti. Strumenti come CodeQL e SonarQube sono fondamentali in questo contesto per l'identificazione di schemi di codice pericolosi e vulnerabilità conosciute. Studi comparativi tra il codice generato dai Large Language Models e le soluzioni umane fornite su piattaforme come Stack Overflow rivelano differenze significative sia nel tipo che nella frequenza degli errori [18]. Questo lavoro evidenzia la necessità di strategie ad hoc per la revisione e correzione del codice sorgente prodotto dai modelli di intelligenza artificiale. L'analisi statica, quindi, è essenziale non solo per individuare errori comuni, ma anche per la sua capacità di adattarsi e rilevare nuove vulnerabilità emergenti con l'avanzamento tecnologico.

2.5 Limiti dello stato dell'arte

Sebbene i precedenti studi sull'applicazione dei Large Language Models abbiano dimostrato la loro efficienza in molti aspetti, hanno anche evidenziato alcuni potenziali problemi significativi riguardanti la sicurezza nella generazione del codice, nel senso che il codice scritto da terze parti dovrebbe essere trattato con molta cautela e

accuratamente rivisto per garantirne la sicurezza [18]. Ovviamente, resta aperta la questione di come si svilupperanno esattamente gli LLM e quale sarà il loro impatto sulla sicurezza del codice generato. Poiché i modelli continueranno a evolversi nel tempo e gli strumenti saranno migliorati per una migliore verifica e validazione, esiste la necessità di una revisione continua degli strumenti di analisi delle vulnerabilità e di mitigazione dei rischi. Questo processo sarà supportato da strumenti e tecniche che rafforzeranno i modelli contro i rischi linguistici su larga scala, consentendo l'analisi e la mitigazione delle vulnerabilità in coevoluzione con lo sviluppo di tali modelli. L'innovazione rapida nell'ambito dell'intelligenza artificiale richiede un adattamento altrettanto rapido nel campo della sicurezza informatica. In questo contesto, gli esperti di IA dovrebbero collaborare strettamente con gli esperti di sicurezza informatica per garantire che i modelli sviluppati non siano solo superiori dal punto di vista funzionale, ma anche sicuri di per sé. Infine, l'attenzione crescente verso le implicazioni di sicurezza degli LLM richiede maggiore consapevolezza e proattività. Allo stesso tempo, serve come promemoria del fatto che i rischi associati al codice generato automaticamente non vanno sottovalutati, e che ulteriori tecnologie e metodi per il suo sviluppo e la sua verifica devono ancora essere sviluppati.

CAPITOLO 3

Metodo di ricerca

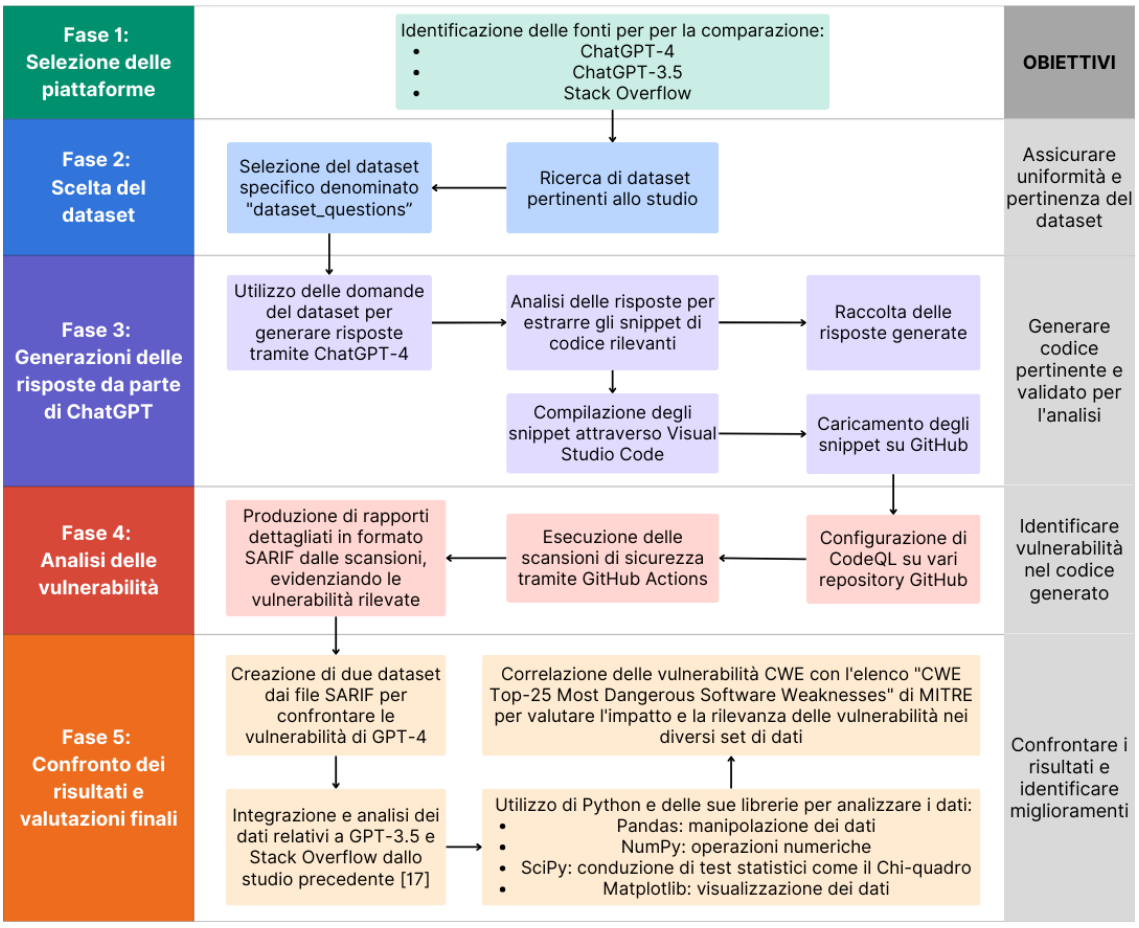

Figura 3.1: Diagramma Metodo di ricerca.

In questo studio sperimentale, il confronto tra le fonti di informazione è stato articolato in cinque fasi principali. La prima fase consiste nella selezione di tre piattaforme da confrontare e analizzare: ChatGPT-4, ChatGPT-3.5 e Stack Overflow. La seconda fase riguarda l'uso di un dataset preesistente di domande e risposte contenenti tematiche di sicurezza su Stack Overflow. Le stesse domande sono state sottoposte a ChatGPT-4 per generare snippet di codice (fase 3), i quali sono stati successivamente raccolti per l'analisi. Nella fase 4, è stata condotta un'analisi delle vulnerabilità sugli snippet generati grazie all'utilizzo di CodeQL. Infine, i risultati ottenuti da ChatGPT-4 sono stati confrontati con quelli di una ricerca precedente [18] che ha analizzato le risposte di ChatGPT-3.5 e StackOverflow (fase 5).

3.1 Fase 1: Selezione delle piattaforme

La selezione delle piattaforme utilizzate nello studio è stata determinata principalmente dagli obiettivi prefissati. Tra i Large Language Models, è stato scelto ChatGPT-4, sviluppato da OpenAI, per vari motivi. In primo luogo, il suo impatto è stato significativo: come riportato dal quotidiano The Guardian [21], ChatGPT ha raggiunto 100 milioni di utenti nei primi due mesi dal lancio, un traguardo più rapido rispetto a piattaforme come TikTok, che ha impiegato circa nove mesi per raggiungere lo stesso numero di utenti, e Instagram, che ha superato i due anni per arrivare alla stessa soglia. Un altro aspetto rilevante di ChatGPT è la possibilità di interagire con esso come con un agente conversazionale, simile al modo in cui uno sviluppatore può fare domande e ricevere risposte su Stack Overflow.

Inoltre, la scelta di ChatGPT-4 è stata motivata dall'opportunità di confrontare il miglioramento del modello rispetto a ChatGPT-3.5, grazie a uno studio preesistente che utilizza lo stesso dataset e metodologia. Questo confronto diretto tra i due modelli offre un'analisi approfondita delle evoluzioni nelle capacità di generazione del codice e nella gestione delle vulnerabilità.

Come fonte di informazioni tradizionale basata sul web, è stato scelto Stack Overflow, in linea con lo studio precedente. Nonostante il suo utilizzo sia in leggero declino, come discusso nel Paragrafo 1.1, rimane una piattaforma di riferimento

ampiamente frequentata dai programmatori per la condivisione di conoscenze e soluzioni.

3.2 Fase 2: Scelta del dataset

Nell'articolo di riferimento [18], l'analisi si basava su tre diversi dataset [22]:

- dataset_questions: Contenente le domande di Stack Overflow utilizzate per l'analisi.
- dataset_answers: Comprendente le risposte selezionate da Stack Overflow.
- dataset_snippets: Che include coppie di frammenti di codice derivati sia da ChatGPT-3.5 sia dalle risposte di Stack Overflow, oggetto di confronto nello studio.

Per questo studio, è stato sufficiente l'utilizzo del dataset "dataset_questions", il quale era già stato accuratamente selezionato e preparato specificamente per le analisi condotte nell'articolo [18]. Questo particolare insieme di dati include le domande estratte da Stack Overflow, specificamente incentrate su problemi di sicurezza informatica, e spesso arricchite da snippet di codice. L'adozione di questo dataset è stata guidata dalla sua pertinenza diretta e applicabilità nell'ambito delle interazioni legate alla sicurezza del software.

Il dataset scelto si distingue per la sua immediatezza e per la qualità delle informazioni che contiene, le quali erano già state sottoposte a un rigoroso processo di selezione e validazione per garantire la loro rilevanza e accuratezza. Questa scelta ha permesso di bypassare ulteriori fasi di selezione o filtraggio dei dati, focalizzando l'attenzione direttamente sull'analisi comparativa con le risposte generate dai Large Language Models.

3.3 Fase 3: Generazione delle risposte da parte di ChatGPT

Dopo aver definito il dataset nella Fase 2, la terza fase dello studio si è concentrata sulla generazione di snippet di codice mediante ChatGPT-4. Questa fase è stata

cruciale per valutare la capacità del modello di linguaggio di generare soluzioni di codice pertinenti e sicure in risposta alle domande tratte da StackOverflow.

Per ogni domanda del dataset 'dataset_questions', è stato interpellato ChatGPT-4, specificando la necessità di generare risposte che includessero codice compilabile completo, piuttosto che semplici frammenti. Questo approccio mirava a simulare il tipo di interazione che uno sviluppatore potrebbe avere in un contesto reale di risoluzione dei problemi.

Le risposte ottenute da ChatGPT-4 sono state minuziosamente esaminate per estrarre gli snippet di codice rilevanti. Questi sono stati successivamente salvati separatamente, facilitando l'analisi in fasi successive [23]. La revisione degli snippet ha assicurato che il codice fosse funzionale e strettamente pertinente alle domande poste, riflettendo scenari autentici di programmazione.

In preparazione per l'analisi successiva, sono stati resi tutti gli snippet compilabili attraverso Visual Studio Code, assicurando che il codice fosse non solo corretto sintatticamente ma anche esente da errori di compilazione. Dopo aver validato la loro funzionalità, sono stati caricati gli snippet su GitHub. Questo non solo ha semplificato la gestione del codice ma ha anche garantito che l'ambiente di test riflettesse condizioni reali di sviluppo, ponendo le basi per una valutazione dettagliata delle vulnerabilità tramite CodeQL.

3.4 Fase 4: Analisi delle vulnerabilità

Con gli snippet di codice ora funzionali e accessibili su GitHub, la fase 4 dello studio si è concentrata sull'identificazione e analisi delle vulnerabilità di sicurezza. Utilizzando CodeQL (vedi Paragrafo 2.3.2) è stato possibile esaminare in modo approfondito gli snippet generati da ChatGPT-4 per scoprire eventuali debolezze o errori di sicurezza.

Inizialmente è stato configurando CodeQL nelle varie repository GitHub. Questo processo ha incluso la definizione di workflow specifici per l'analisi del linguaggio di programmazione dei nostri snippet. CodeQL analizza il codice sorgente trasformandolo in un database di query, che poi viene esplorato per identificare modelli corrispondenti a vulnerabilità conosciute.

Le scansioni di sicurezza, automatizzate attraverso GitHub Actions, sono eseguite automaticamente ogni volta che modifiche al codice venivano caricate nelle repository. Questo ha permesso di avere un controllo continuo e aggiornato delle vulnerabilità durante tutto il processo di sviluppo. Ogni scansione di sicurezza ha prodotto un rapporto dettagliato in formato SARIF, elencando le vulnerabilità rilevate, categorizzate in base al loro livello di rischio. Questo formato di file standardizzato facilita l'analisi e l'integrazione dei dati con altri strumenti di sicurezza, permettendo una valutazione approfondita e sistematica delle minacce al codice.

3.5 Fase 5: Confronto dei risultati e valutazioni finali

Nella quinta e conclusiva fase dello studio, è stata condotta un'analisi comparativa dettagliata delle vulnerabilità rilevate nei codici generati da ChatGPT-4 rispetto a quelle presenti nei codici manuali di Stack Overflow e in quelli prodotti dalla versione precedente del modello, ChatGPT-3.5. Questa comparazione ha permesso di esaminare l'evoluzione della qualità e della sicurezza del codice prodotto dai modelli di linguaggio avanzati.

Per facilitare questo confronto, sono stati creati due dataset distinti utilizzando i rapporti in formato SARIF generati da CodeQL [23]. Il primo dataset raccoglieva le vulnerabilità identificate nei codici generati da ChatGPT-4, includendo dettagli come il tipo di vulnerabilità e una descrizione specifica del problema individuato. Questo ha permesso di esaminare in dettaglio le vulnerabilità uniche introdotte dal Large Language Model.

Il secondo dataset, invece, include i tipi di CWE (Common Weakness Enumeration) rilevati per ogni risposta di ChatGPT-4, insieme al livello di gravità associato a ciascuna vulnerabilità identificata. Questo approccio ha consentito di valutare non solo la quantità, ma anche l'importanza e la criticità delle vulnerabilità riscontrate, offrendo una prospettiva dettagliata sull'affidabilità del modello nel generare codice sicuro.

Sono state utilizzate un insieme standardizzato di metriche per valutare e confrontare le vulnerabilità tra i diversi set di dati. Queste metriche includevano il numero di vulnerabilità per tipo, la gravità delle stesse, e la frequenza di particolari errori

di sicurezza. L'obiettivo era identificare miglioramenti o regressi nella capacità dei Large Language Models di generare codice sicuro.

Per garantire la validità statistica dei risultati, è stato impiegato il linguaggio di programmazione Python, utilizzando librerie come Pandas per la manipolazione dei dati, NumPy per le operazioni numeriche, SciPy per condurre il test del Chi-quadro, e Matplotlib per la visualizzazione dei dati. Questi strumenti hanno facilitato l'analisi dettagliata della distribuzione delle frequenze delle vulnerabilità. Inoltre, sono state correlate le CWE identificate con l'elenco "CWE Top-25 Most Dangerous Software Weaknesses" del MITRE [8], consentendo un'ulteriore valutazione sull'impatto e sulla rilevanza delle vulnerabilità nei vari set di dati.

CAPITOLO 4

Analisi dei risultati

Tabella 4.1: Sintesi delle vulnerabilità rilevate in ChatGPT e Stack Overflow. È evidenziata la piattaforma con meno vulnerabilità.

	GPT-4	**GPT-3.5**	**Stack Overflow**
Snippet di codice con vulnerabilità	49	87	83
Vulnerabilità negli snippet	101	248	302

Nota: *Alcuni dati presentati in questa tabella sono stati ricavati dall'articolo [18].*

4.1 Vulnerabilità

Per esaminare le evoluzioni nella sicurezza dei modelli di linguaggio e il loro confronto con le risposte umane su Stack Overflow, sono stati analizzati gli snippet di codice per la presenza di vulnerabilità utilizzando CodeQL. È importante notare che l'analisi per GPT-4 è stata condotta su 87 snippet, mentre per GPT-3.5 e Stack Overflow su 108 snippet ciascuno. Su 87 snippet analizzati, GPT-4 ha generato 49 snippet con vulnerabilità. Sebbene questo risultato mostri un miglioramento rispetto agli 87 snippet vulnerabili su 108 di GPT-3.5 e agli 83 di Stack Overflow, l'interpre-

tazione deve considerare la diversa dimensione del campione. GPT-4 ha mostrato un totale di 101 vulnerabilità, significativamente inferiore alle 248 vulnerabilità in GPT-3.5 e alle 302 in Stack Overflow, anche se su un numero inferiore di snippet. Questo indica una densità inferiore di vulnerabilità per snippet in GPT-4 rispetto agli altri, suggerendo miglioramenti nella generazione del codice.

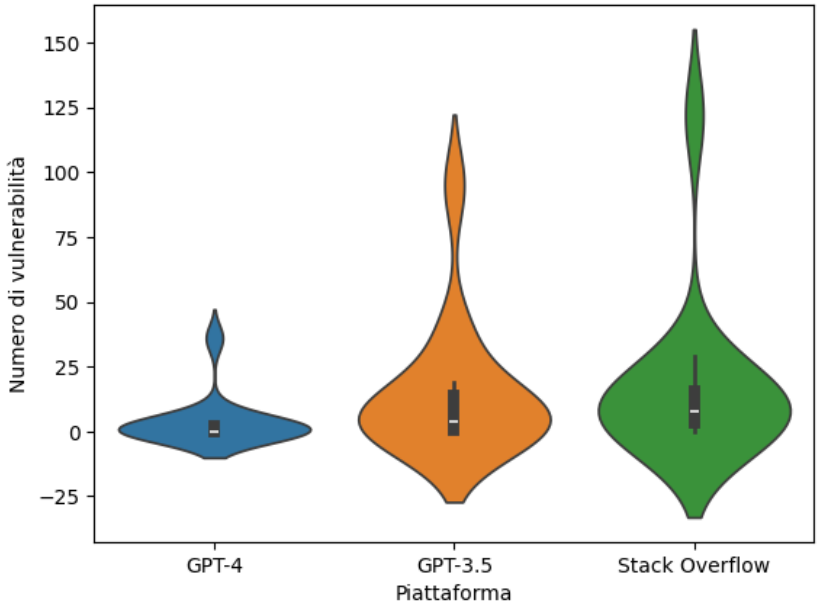

Figura 4.1: Numero di vulnerabilità negli snippet di codice in ogni piattaforma.

La Figura 4.1 aggiornata mostra un grafico a violino integrato con un box plot che rappresenta la distribuzione del numero di vulnerabilità per gli snippet di codice generati da GPT-4, GPT-3.5, e Stack Overflow. La distribuzione per GPT-4 mostra un'ampiezza minore rispetto alle altre piattaforme, indicando una minor variabilità nel numero di vulnerabilità per snippet. Il grafico suggerisce che GPT-4 tende a generare snippet con un numero relativamente basso e consistente di vulnerabilità. La mediana, indicata dalla linea nera nel box plot, si posiziona vicino al limite inferiore della distribuzione, suggerendo che la maggior parte degli snippet di GPT-4 contiene poche vulnerabilità. In confronto, GPT-3.5 e Stack Overflow mostrano distribuzioni più ampie e mediane più elevate. Questo indica che, sebbene possano generare snippet con un numero minore di vulnerabilità in alcuni casi, tendono anche a produrre snippet con un numero significativamente maggiore di vulnerabilità in altri. La presenza di code lunghe nel grafico a violino di Stack Overflow suggerisce episodi in cui gli snippet hanno un numero eccessivamente alto di vulnerabilità.

4.2 Vulnerabilità CWE

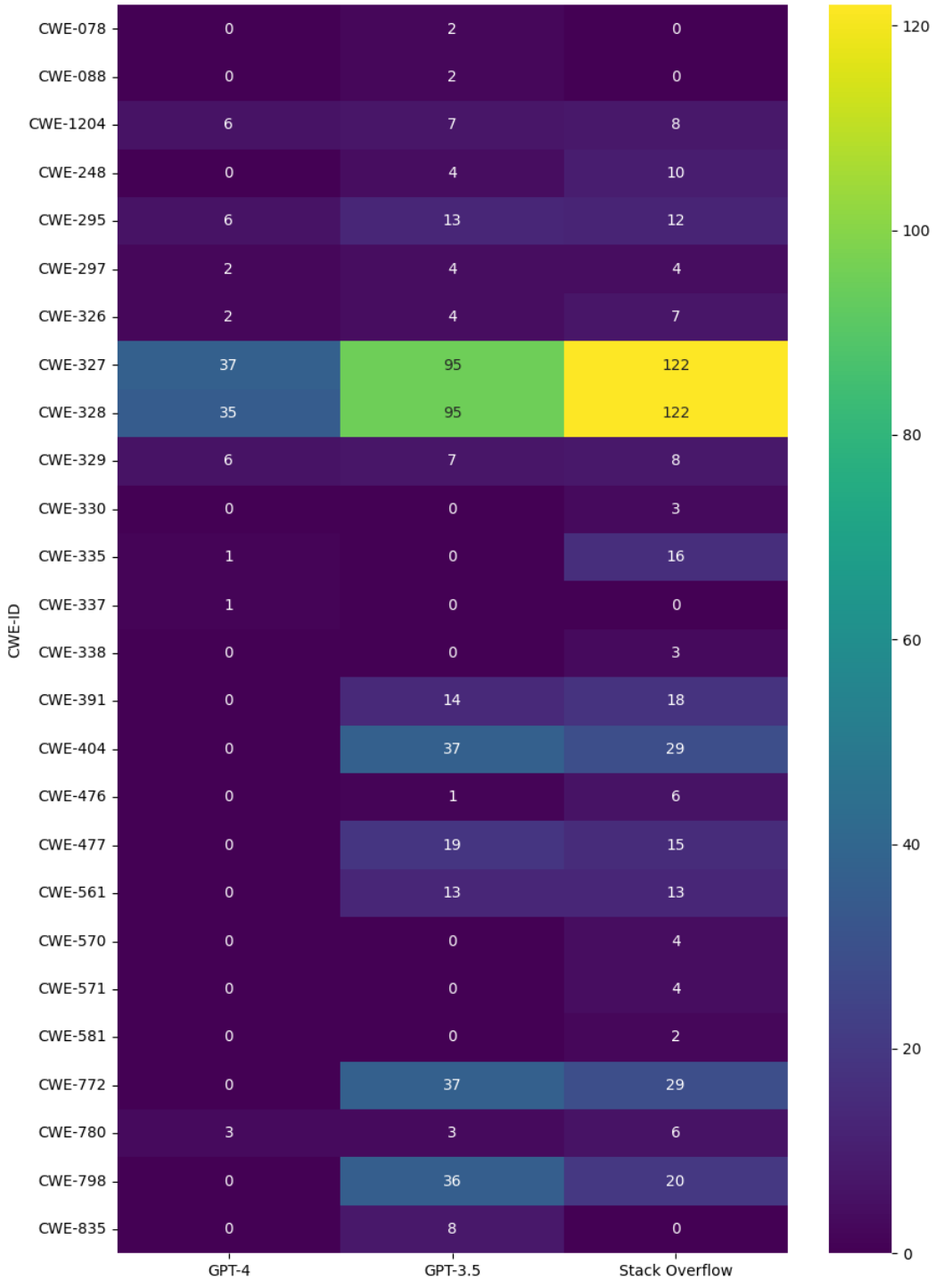

Figura 4.2: Distribuzione dei tipi di CWE rilevati nelle diverse piattaforme.

Nota: *Alcuni dati presentati in questa figura sono stati ricavati dall'articolo [18].*

Tabella 4.2: Numero totale di CWE rilevate nelle diverse piattaforme.

	GPT-4	**GPT-3.5**	**Stack Overflow**
CWE	10	19	22

Nota: *Alcuni dati presentati in questa tabella sono stati ricavati dall'articolo [18].*

Nel modello ChatGPT-4 sono state trovate 10 vulnerabilità CWE. Il tipo più ricorrente è il CWE-327 relativo all'uso di algoritmi crittografici obsoleti o pericolosie, seguito dal CWE-328 che riguarda l'impiego di hash deboli. Questi errori sono rilevati dalle regole di CodeQL "Uso di un algoritmo crittografico rotto o rischioso" e "Uso potenzialmente pericoloso di un algoritmo crittografico".

Tra gli altri CWE rilevati frequentemente ci sono il CWE-1204, che riguarda l'uso inappropriato di dati sensibili, il CWE-295, che indica la mancanza di validazione dei certificati SSL/TLS, e il CWE-329, legato all'uso non sicuro di algoritmi di crittografia con chiavi insufficienti, con ciascuno di questi presenti in 6 snippet per un totale di 18. Tra i CWE meno frequenti figurano il CWE-297, che tratta il controllo improprio dell'autenticità di un certificato, il CWE-326, relativo all'adeguatezza dei meccanismi di cifratura, il CWE-335, che indica l'uso non corretto dei semi nel generatore di numeri pseudorandom (PRNG), il CWE-337, sulla prevedibilità nella generazione di numeri casuali, e il CWE-780, legato all'uso non sicuro di algoritmi per l'autenticazione.

Per quanto riguarda le vulnerabilità più frequenti tra le diverse piattaforme, emergono il CWE-327 e CWE-328. GPT-4 ha mostrato un miglioramento nella gestione delle vulnerabilità legate all'uso di algoritmi crittografici rischiosi e hash deboli, mentre per GPT-3.5 e Stack Overflow, il volume di queste vulnerabilità rimane elevato, sottolineando che questi problemi sono comuni e persistenti nell'ambiente di sviluppo software. Per il CWE-404, che riguarda la gestione impropria delle risorse, si nota che è particolarmente problematica in GPT-3.5 e Stack Overflow, con rispettivamente 37 e 29 occorrenze. Questo indica che i problemi di gestione delle risorse rappresentano un problema significativo e ricorrente sia nei modelli di linguaggio che nelle pratiche di programmazione umana. Il CWE-335 emerge principalmente in Stack Overflow con 16 occorrenze, riflettendo una specifica sfida nell'ambito della

programmazione umana che non è stata altrettanto presente nei modelli GPT. Il CWE-798, relativo all'uso di credenziali hard-coded, è più prominente in GPT-3.5 e Stack Overflow, rispettivamente con 36 e 20 occorrenze. La persistenza di questa vulnerabilità in ambienti di programmazione avanzati come i modelli GPT sottolinea l'importanza di integrare migliori pratiche di sicurezza nei processi di addestramento e generazione del modello.

In GPT-3.5 sono stati rilevati 19 CWE, invece, in Stack Overflow 22. Pertanto, GPT-4 ha generato meno CWE in confronto alle altre piattaforme. È interessante notare che l'ordine delle CWE più comuni è simile per tutte le piattaforme, con un miglioramento più netto di GPT-4, dove alcuni tipi di CWE tendono a scomparire. Per i CWE principali, GPT-4 e GPT-3.5 hanno creato meno vulnerabilità per CWE-327 e CWE-328. Nel frattempo, in GPT-4 sono scompare vulnerabilità come: CWE-708, CWE-088, CWE-248, CWE-391, CWE-404, CWE-476, CWE-477, CWE-561, CWE-772, CWE-798, CWE-835.

Gli snippet di ChatGPT-4 hanno mostrato prestazioni migliori rispetto a quelli di ChatGPT-3.5 per 18 tipi di CWE, mentre ChatGPT-3.5 ha superato ChatGPT-4 in 2 tipi di CWE. Inoltre, per sei tipi di CWE, entrambe le versioni hanno registrato lo stesso numero di snippet. Le differenze più significative sono state osservate per i CWE-327 e CWE-328, con una discrepanza rispettivamente di 58 e 60 occorrenze. Anche i CWE-772 e CWE-404 hanno mostrato grandi divergenze, con 37 occorrenze ciascuno, mentre i CWE-391, CWE-477, CWE-561 e CWE-798 hanno registrato 14, 19, 13 e 36 occorrenze. Le altre discrepanze variano da 0 a 8 occorrenze.

Rispetto a Stack Overflow, gli snippet di ChatGPT-4 sono risultati migliori per 22 tipi di CWE, mentre Stack Overflow ha registrato migliori risultati per un solo tipo di CWE. Per 3 tipi di CWE, le occorrenze sono state equivalenti. I CWE con le maggiori differenze includono tutti quelli già identificati nel confronto con GPT-3.5, con l'aggiunta dei CWE-248 e CWE-335. In particolare, le differenze per i CWE-327 e CWE-328 sono state di 85 e 87 occorrenze, per i CWE-772 e CWE-404 di 29, per il CWE-477 e CWE-335 di 15, e per i CWE-248, CWE-391, CWE-561 e CWE-798 di 10, 18, 13 e 20 rispettivamente. Le altre discrepanze variano anch'esse da 0 a 8 occorrenze.

4.2.1 Analisi della significatività statistica delle CWE tra GPT-4 e Stack Overflow

È stata esaminata la significatività statistica delle differenze nelle frequenze di varie CWE tra GPT-4 e Stack Overflow utilizzando il test Chi-quadro. Le analisi hanno rivelato che alcune CWE presentano una significatività statistica notevole, suggerendo differenze sostanziali nella generazione di vulnerabilità tra le due piattaforme.

1. **CWE-327 "Uso di un algoritmo crittografico rotto o rischioso"**:
    - GPT-4 ha generato questa vulnerabilità 37 volte, mentre Stack Overflow l'ha presentata ben 122 volte, dimostrando una differenza altamente significativa (Chi2=45.44, p-value=1.57e-11).
2. **CWE-328 "Uso di un hash debole"**:
    - Similmente, GPT-4 ha registrato 35 occorrenze, confrontate con 122 in Stack Overflow, con una differenza anch'essa molto significativa (Chi2=48.21, p-value=3.83e-12).
3. **CWE-404 "Arresto o rilascio improprio di una risorsa"**:
    - Non presente in GPT-4, mentre in Stack Overflow si sono verificate 29 occorrenze, segnalando una notevole differenza (Chi2=29.00, p-value=7.24e-08).
4. **CWE-772 "Rilascio mancante di una risorsa dopo la durata effettiva"**:
    - Similmente alla CWE-404, non presente in GPT-4, mentre in Stack Overflow troviamo 29 occorrenze (Chi2=29.0, p-value=7.24e-08).
5. **CWE-798 "Uso di credenziali hard-coded"**:
    - In GPT-4 non presente, ma in Stack Overflow il problema è ancora presenze con 20 occorrenze (Chi2=20.0, p-value=7.74e-06).
6. **CWE-391 "Uso errato di componenti software"**:

- La non presenza di questa CWE in GPT-4 mostra che il modello evita errori di implementazione software che sono invece riscontrati in Stack Overflow, con 14 occorrenze (Chi2=18.0, p-value=2.21e-05).

4.2.2 Analisi della significatività statistica delle CWE tra GPT-4 e GPT-3.5

Sono state confrontate anche le frequenze delle CWE tra GPT-4 e la sua versione precedente, GPT-3.5, per identificare miglioramenti o regressioni nella gestione delle vulnerabilità.

1. **CWE-327 "Uso di un algoritmo crittografico rotto o rischioso"**:
   - È stata registrata 37 volte in GPT-4 rispetto a 95 in GPT-3.5 (Chi2=25.48, p-value=4.46e-07).
2. **CWE-328 "Uso di un hash debole"**:
   - Ci sono state 35 occorrenze in GPT-4 contro 95 in GPT-3.5 (Chi2=27.69, p-value=1.42e-07), indicando miglioramenti significativi in GPT-4.
3. **CWE-404 "Arresto o rilascio improprio di una risorsa"**:
   - Questa vulnerabilità, non registrata in GPT-4, ha avuto 37 occorrenze in GPT-3.5, evidenziando un miglioramento sostanziale (Chi2=37.00, p-value=1.18e-09).
4. **CWE-772 "Rilascio mancante di una risorsa dopo la durata effettiva"**:
   - L'assenza di queste vulnerabilità in GPT-4 rispetto a GPT-3.5 (37 occorrenze) dimostra un miglioramento sostanziale nella gestione delle risorse, confermando la maturità crescente del modello (Chi2=37.0, p-value=1.18e-09).
5. **CWE-798 "Uso di credenziali hard-coded"**:

- Ancora una volta, un miglioramento notevole è stato osservato con zero occorrenze in GPT-4 rispetto a 36 in GPT-3.5 (Chi2=36.00, p-value=1.97e-09).

6. **CWE-477 "Uso errato di un'API"**:

   - Un miglioramento significativo in GPT-4 (0 occorrenze) mostra che il modello è più capace di interfacciarsi correttamente con le API, rispetto a GPT-3.5 (19 occorrenze), riducendo gli errori di implementazione (Chi2=19.0, p-value=1.31e-05).

4.2.3 Risultati rilevati dall'analisi statistica

L'applicazione del test del Chi-quadro ha rivelato risultati significativi che evidenziano le differenze nella generazione di vulnerabilità tra le diverse piattaforme.

- **Confronto tra ChatGPT-4 e Stack Overflow:** La frequenza di alcune CWE tra ChatGPT-4 e Stack Overflow mostra differenze statisticamente significative. Ad esempio, la CWE-327 è stata generata meno frequentemente da ChatGPT-4 rispetto a Stack Overflow, indicando una possibile maggiore attenzione del modello nella scelta di algoritmi crittografici più sicuri. Similmente, la non presenza di vulnerabilità come la CWE-404 e CWE-798 in ChatGPT-4 rispetto alle loro presenze in Stack Overflow suggerisce che il modello evita alcuni errori comuni di gestione delle risorse e sicurezza delle credenziali.

- **Confronto tra ChatGPT-4 e ChatGPT-3.5:** L'analisi ha mostrato miglioramenti significativi nella generazione di codice sicuro da parte di ChatGPT-4 rispetto alla sua versione precedente, ChatGPT-3.5. Ad esempio, l'assenza di vulnerabilità come la CWE-404 e CWE-798 in ChatGPT-4, che erano presenti in ChatGPT-3.5, dimostra un avanzamento nella capacità del modello di evitare errori critici di gestione delle risorse e di sicurezza. Inoltre, la riduzione delle occorrenze delle CWE-327 e CWE-328 rafforza l'indicazione di un miglioramento nella robustezza del codice generato.

Questi risultati forniscono una base empirica per discutere la progressione nella sicurezza dei modelli di linguaggio su diverse piattaforme e versioni, evidenziando aree di forza e di miglioramento.

4.3 Analisi top 25 CWE del MITRE

Analizzando la Top 25 CWE del MITRE per il 2023 [8], emerge che GPT-4 non manifesta alcuna delle vulnerabilità elencate in questa classifica, a differenza di GPT-3.5 e Stack Overflow. Tra le vulnerabilità identificate in questi ultimi, due rientrano nella Top 25 del MITRE. La prima, il CWE-476 "NULL Pointer Dereference", che si classifica al 12° posto, descrive una situazione in cui un software tenta di utilizzare un puntatore che si aspetta essere valido, ma che in realtà è nullo; questa vulnerabilità è stata rilevata in un frammento di codice di GPT-3.5 e in sei frammenti di Stack Overflow. La seconda, il CWE-798 "Use of Hard-coded Credentials", si trova al 18° posto e riguarda l'uso di credenziali codificate direttamente nel software; è stata osservata questa vulnerabilità 36 volte negli snippet di GPT-3.5 e 20 volte in quelli di Stack Overflow.

CAPITOLO 5

Conclusioni

5.1 Riepilogo del lavoro svolto

Il presente lavoro di tesi si è proposto di confrontare le vulnerabilità di sicurezza presenti nel codice generato da GPT-4 con quello generato da GPT-3.5 e le risposte di Stack Overflow. Utilizzando un dataset di domande sulla sicurezza provenienti da Stack Overflow, è stata condotta un'analisi comparativa del codice generato da questi tre sistemi mediante lo strumento di analisi statica CodeQL, identificando e classificando le vulnerabilità secondo il Common Weakness Enumeration (CWE).

L'obiettivo principale è stato valutare l'affidabilità dei Large Language Models nella generazione di codice sicuro, con particolare attenzione alle differenze tra GPT-3.5, GPT-4 e Stack Overflow, per comprendere come l'evoluzione dei modelli e l'uso di fonti tradizionali influenzino la sicurezza del codice prodotto.

5.2 Risultati principali

L'analisi ha evidenziato che GPT-4 ha mostrato significativi miglioramenti rispetto a GPT-3.5 in termini di riduzione del numero di vulnerabilità. In particolare, su 87 snippet analizzati, GPT-4 ha prodotto un totale di 101 vulnerabilità, contro le 248

di GPT-3.5 e le 302 di Stack Overflow, dimostrando una maggiore robustezza nella generazione di codice sicuro. Inoltre, GPT-4 ha presentato solo 10 tipi di CWE, rispetto ai 19 di GPT-3.5 e ai 22 di Stack Overflow, suggerendo una riduzione della ripetitività degli errori di sicurezza.

Le vulnerabilità più comuni rilevate in GPT-4 sono state il CWE-327 (uso di algoritmi crittografici obsoleti) e il CWE-328 (hash deboli), in linea con quanto riscontrato in GPT-3.5 e Stack Overflow, sebbene con una frequenza inferiore. In particolare, GPT-4 ha migliorato significativamente la gestione di vulnerabilità legate all'uso di algoritmi crittografici rischiosi e hash deboli, con un numero inferiore di occorrenze rispetto agli altri due sistemi. Questi risultati suggeriscono che GPT-4 è in grado di generare codice con una densità inferiore di vulnerabilità per snippet rispetto ai suoi predecessori.

5.3 Impatto del lavoro

Il contributo di questa tesi è di grande rilevanza per la comunità di sviluppatori e ricercatori impegnati nell'uso di Large Language Models (LLM) per la generazione automatica di codice. L'analisi condotta dimostra che evoluzioni recenti in modelli come GPT-4 hanno notevolmente migliorato la sicurezza del codice prodotto, mitigando significativamente il rischio di vulnerabilità critiche rispetto alle versioni precedenti e alle soluzioni frequentemente suggerite su Stack Overflow. Nonostante questi progressi, alcune vulnerabilità persistenti, come l'uso di algoritmi crittografici obsoleti e la gestione inadeguata delle eccezioni, richiedono un'attenzione continua. Questi risultati evidenziano un'evoluzione nel ruolo del programmatore, che si sposta sempre più verso un supervisore della qualità del codice generato automaticamente, enfatizzando la necessità di una solida comprensione delle pratiche di sicurezza informatica. Inoltre, la persistenza di tali vulnerabilità solleva questioni significative riguardo alla futura affidabilità degli LLM in ambito di sicurezza: sebbene il progresso sia evidente, la strada verso un'IA generativa completamente affidabile richiede ulteriori miglioramenti e un impegno costante.

Questa tesi fornisce quindi spunti preziosi per sensibilizzare gli sviluppatori sull'adozione di buone pratiche di sicurezza, indispensabili per minimizzare i rischi

associati alla generazione di codice automatizzata e per rispondere efficacemente alla domanda se l'AI generativa potrà mai essere considerata completamente affidabile in ambito di sicurezza.

5.4 Limiti del lavoro

Nonostante i risultati promettenti, il presente studio presenta alcune limitazioni significative. Primo, l'analisi è stata condotta esclusivamente su codice Java, il che limita la generalizzabilità dei risultati ad altri linguaggi di programmazione, che potrebbero comportarsi diversamente in termini di sicurezza e vulnerabilità. Inoltre, essendo basata su un dataset specifico di domande provenienti da Stack Overflow, l'analisi potrebbe non coprire l'intero spettro di vulnerabilità che si possono presentare nel codice utilizzato in scenari applicativi reali e diversificati. Infine, gli strumenti di analisi statica adottati, come CodeQL, sebbene efficaci nell'identificare vulnerabilità note, potrebbero non essere adeguati per rilevare nuove forme di vulnerabilità o problemi di sicurezza emergenti e più sofisticati.

5.5 Sviluppi futuri

In considerazione degli esiti di questo studio, vi sono diverse direzioni promettenti per estendere il lavoro futuro. Una possibile evoluzione riguarda l'espansione dell'analisi a linguaggi di programmazione diversi dal Java, quali Python o C++, per determinare se i risultati si mantengano consistenti attraverso diversi ambienti di sviluppo. Un'altra potenziale estensione del lavoro potrebbe includere l'uso di strumenti di analisi dinamica, che possono scoprire vulnerabilità che sfuggono alla sola analisi statica. Inoltre, sarà fondamentale monitorare l'evoluzione dei modelli di linguaggio, come l'ipotetico GPT-5, per valutare se i progressi nei modelli continuino a contribuire alla riduzione delle vulnerabilità di sicurezza nel codice generato, approfondendo così la nostra comprensione dell'interazione tra AI generativa e pratiche di sicurezza informatica.

Bibliografia

[1] A. Hartwell, "Software can have a positive impact on society. here's how to make it happen." *Capita*, 2021. (Citato alle pagine 1 e 2)

[2] R. Kazman and L. Pasquale, "Software engineering in society," *IEEE Software*, vol. 37, no. 1, pp. 7–9, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8938101 (Citato alle pagine 1 e 2)

[3] T. TechinPost, "The importance of software in the modern world," *TechinPost*, 2023. [Online]. Available: https://www.techinpost.com/importance-of-software-in-modern-world/ (Citato a pagina 2)

[4] "Owasp top ten," *OWASP Foundation*, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/ (Citato a pagina 2)

[5] R. KA, "Understanding the owasp top 10 2021 application security risks," *TheSecMaster*, 2024. [Online]. Available: https://thesecmaster.com/blog/understanding-the-owasp-top-10-2021-application-security-risks (Citato a pagina 2)

[6] B. Wang, "Most common types of cyber vulnerabilities," *CrowdStrike*, 2022. [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/cyber-vulnerabilities/ (Citato a pagina 3)

[7] T. Dang, "Common software security issues strategies to prevent their impact," *Orient Software*, 2023. [Online]. Available: https://www.orientsoftware.com/blog/software-security-issues/ (Citato a pagina 3)

[8] "Cwe top 25 most dangerous software weaknesses," *MITRE*, 2023. [Online]. Available: https://cwe.mitre.org/top25/ (Citato alle pagine 3, 21 e 30)

[9] A. Morena, "La lenta agonia di stack overflow, o no?" *codemotion*, 2024. [Online]. Available: https://www.codemotion.com/magazine/it/intelligenza-artificiale/la-lenta-agonia-di-stack-overflow-o-no/ (Citato a pagina 3)

[10] M. R. D. Porta, "Intelligenza artificiale: mercato in forte crescita. accelera l'ia generativa," *i-com*, 2024. [Online]. Available: https://www.i-com.it/2024/04/19/intelligenza-artificiale-mercato-in-forte-crescita-accelera-lia-generativa/ (Citato a pagina 4)

[11] S. Coyne, K. Sakaguchi, D. Galvan-Sosa, M. Zock, and K. Inui, "Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction," *Tohoku University*, 2023. [Online]. Available: https://arxiv.org/pdf/2303.14342 (Citato a pagina 6)

[12] M. Fu, C. K. Tantithamthavorn, V. Nguyen, and T. Le, "Chatgpt for vulnerability detection, classification, and repair: How far are we?" *Monash University*, 2023. [Online]. Available: https://arxiv.org/pdf/2310.09810 (Citato a pagina 6)

[13] "Cos'è un large language model (llm)?" *Cloudflare*, 2023. [Online]. Available: https://www.cloudflare.com/it-it/learning/ai/what-is-large-language-model/ (Citato a pagina 9)

[14] G. Romanato, "La storia e l'evoluzione dei large language models (llm)," *Gabriele Romanato*, 2024. [Online]. Available: https://gabrieleromanato.com/2024/07/la-storia-e-evoluzione-dei-large-language-models-llm (Citato a pagina 9)

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Papers With Code*, 2017. [Online]. Available: https://paperswithcode.com/paper/attention-is-all-you-need (Citato a pagina 9)

[16] WEBMASTER, "Large language models (llm), cosa sono e come funzionano," *Meccanismo Complesso*, 2023. [Online]. Available: https://www.meccanismocomplesso.org/large-language-models-llm/ (Citato a pagina 11)

[17] I. Education, "Software per la generazione di codice ai: cos'è e come funziona?" *IBM*, 2023. [Online]. Available: https://www.ibm.com/it-it/think/topics/ai-code-generation (Citato a pagina 12)

[18] S. Hamer, M. d'Amorim, and L. Williams, "Just another copy and paste? comparing the security vulnerabilities of chatgpt generated code and stackoverflow answers," in *2024 IEEE Security and Privacy Workshops (SPW)*, 2024, pp. 87–94. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10579524 (Citato alle pagine 13, 14, 15, 17, 18, 22, 24 e 25)

[19] "Enumerazione delle debolezze comuni (cwe)," *FasterCapital*, 2024. [Online]. Available: https://fastercapital.com/it/tema/enumerazione-delle-debolezze-comuni-(cwe).html (Citato a pagina 13)

[20] "Analisi del codice con github codeql," *Microsoft*, 2023. [Online]. Available: https://learn.microsoft.com/it-it/training/modules/code-scanning-with-github-codeql/ (Citato a pagina 14)

[21] "Chatgpt reaches 100 million users two months after launch," *The Guardian*, 2023. [Online]. Available: https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app (Citato a pagina 17)

[22] S. Hamer, M. d'Amorim, and L. Williams, "Just another copy and paste? comparing the security vulnerabilities of chatgpt generated code and stackoverflow answers," *Zenodo*, 2024. [Online]. Available: https://zenodo.org/records/10806611 (Citato a pagina 18)

[23] C. Paciello, "Dataset analisi delle vulnerabilità," 2024. [Online]. Available: https://github.com/cost-02/TesiTriennale_CostantinoPaciello (Citato alle pagine 19 e 20)

Ringraziamenti

Desidero innanzitutto ringraziare il mio relatore, Professore **Fabio Palomba**, per la sua costante disponibilità e il supporto che mi ha offerto durante tutto il percorso di questa tesi.

E ora, un ringraziamento speciale va a chi ha segnato più di tutti il mio cammino: i miei genitori. Mamma e papà, grazie per tutto ciò che avete fatto e continuate a fare per me. Ogni vostro gesto, anche il più semplice, mi ricorda quanto sia prezioso il vostro amore.

Mamma, sei stata la mia prima, più grande sostenitrice, sempre pronta a credere in me, anche quando io stesso faticavo a farlo. Prima di ogni esame mi ripetevi: "*Comunque vada, sarà un successo*". All'inizio quella frase mi dava fastidio, forse perché cercavo certezze diverse, o perché non riuscivo a vedere oltre la mia ansia. Ma col tempo ho compreso che quella frase era il tuo modo di regalarmi tranquillità, di infondere pace quando più ne avevo bisogno. Quella frase è diventata un abbraccio silenzioso, una carezza invisibile che mi ha insegnato a guardare le difficoltà con uno spirito più sereno. Crescendo, mi accorgo di quanto io abbia assimilato una parte del tuo carattere: la tua precisione, la tua riflessività e quella determinazione incrollabile, accompagnata da una vena polemica che ora riconosco sempre più spesso in me. Questi tratti sono diventati una parte di me, e per questo ti sarò eternamente grato.

Papà, con i tuoi discorsi e le tue riflessioni filosofiche, mi hai insegnato a guardare il

mondo con curiosità e mente aperta. Ma spesso, più delle parole, erano i tuoi sguardi a parlare: intensi, silenziosi, capaci di trasmettere ciò che mille discorsi non avrebbero potuto dire. Da te ho ereditato l'ingegno, la voglia di capire e di andare oltre, e quel desiderio costante di scoprire qualcosa di nuovo ogni giorno, un passo alla volta. Grazie per avermi mostrato che la vera conoscenza non si esaurisce mai, ma cresce con la passione e l'attenzione verso ciò che ci circonda.
Anche nei momenti più difficili, ho sempre sentito il calore del vostro sostegno accanto a me. So di essere stato a volte distante, e mi dispiace, ma spero sappiate quanto vi voglio bene. Non basteranno mai le parole per dirvi grazie: vi sarò grato per sempre.

A mio **fratello**, grazie per avermi sempre supportato e, spesso, anche sopportato. Sei stato, e continui a essere, un punto di riferimento importante nella mia vita, qualcuno su cui so di poter sempre contare. La tua capacità di farmi vedere le cose da una prospettiva diversa è stata un aiuto fondamentale lungo il mio percorso. Hai saputo darmi forza e motivazione quando ne avevo più bisogno, ma la cosa che più apprezzo è la leggerezza che porti nella mia vita. Quante volte abbiamo riso fino alle lacrime per battute che probabilmente non avevano neanche senso, ma che in quel momento erano tutto ciò di cui avevo bisogno. Quei momenti sono stati un modo per lasciarmi alle spalle le preoccupazioni, anche solo per un po'. Grazie per esserci sempre stato. Non potrò mai esprimere a parole quanto significhi per me.

Non posso dimenticare i miei nonni: Costantino, Michela, Giuseppe e Carolina. Anche se non siete più fisicamente accanto a me, il vostro calore mi avvolge ogni giorno. Mi rattrista che non possiate essere presenti in questo momento speciale, ma sono certo che, da lassù, mi guardate con occhi pieni d'orgoglio e con il cuore colmo di amore.
Nonno Costantino, le nostre uscite in macchina, ad ascoltare le canzoni di Tony Tammaro mentre andavamo alla ricerca della casa dell'Orco, resteranno sempre tra i miei ricordi più belli. Mi hai insegnato che, per quanto la vita possa essere dura, non bisogna mai arrendersi. Dopo quella brutta operazione non riuscivo a vederti o a parlarti come prima, e questo mi faceva male. Hai affrontato tante battaglie, ma, nonostante ciò, hai sempre trovato il modo di farmi sorridere, dimostrando una forza

e un amore straordinari.
Nonna Michela, la tua casa era un luogo magico per me. Ricordo i tuoi impasti in cucina, il profumo dei dolci e la cura con cui preparavi tutto per noi. Ogni momento con te era pieno di calore e amore, e il tuo sorriso illuminava ogni cosa.
Nonno Giuseppe, mi dispiace non averti conosciuto meglio. Non ho tanti ricordi, ma quella foto di noi due sulle scale conta tanto per me: in quell'immagine sento l'amore che mi trasmettevi, e quel sentimento vive ancora dentro di me.
Nonna Carolina, ogni visita da te era una scoperta. I tuoi racconti pieni di vita e le giornate passate a cucinare gli impasti sul fuoco con Alessia erano momenti semplici, ma straordinari. Ogni istante passato con te è un ricordo che custodisco con affetto.

Un sincero riconoscimento va ai miei **zii** e **zie**. Siete sempre stati pronti a mettervi a disposizione nei momenti di necessità, offrendo il vostro affetto, consigli e parole di conforto. Ogni incontro, ogni piccolo gesto di gentilezza, mi ha fatto sentire supportato e mai solo. Il vostro amore e la vostra presenza hanno significato tanto per me.

Un ringraziamento speciale va anche ai miei **cugini** e alle mie **cugine**. La vostra compagnia ha reso questo percorso più facile e pieno di momenti felici. Le risate, le chiacchierate infinite e le giornate spensierate che abbiamo passato insieme sono ricordi che porterò sempre nel cuore. Ognuno di voi ha lasciato un segno speciale in questo viaggio, e i ricordi che abbiamo costruito insieme saranno sempre con me.

Un grazie sentito va ad Andrea, Vittorio e Gaetano, alias GruppoDeiBro, con cui ho vissuto questi anni universitari indimenticabili. La vostra presenza è stata talmente importante che, anche nei momenti di maggiore ansia per gli esami e di noia per alcune lezioni, siete riusciti a distogliere la mia attenzione, trasformando ogni difficoltà in qualcosa di più leggero. Siete riusciti a prendere quelle emozioni negative e a trasformarle in momenti piacevoli, regalandomi risate e spensieratezza che non dimenticherò mai.
Andrea, con cui ho condiviso gli anni delle superiori e dell'università, vivendo insieme sfide, risate e momenti di crescita. In questi anni, ci siamo visti cambiare, imparando l'uno dall'altro a ogni passo e costruendo ricordi che rimarranno per

sempre.
Vittorio, mi hai sempre incoraggiato e dato una mano in ogni momento, rendendo tutte le difficoltà più facili da affrontare. Le tue parole di incoraggiamento e le pacche sulle spalle sono state un sostegno che mi ha dato forza. Mi hanno aiutato a superare gli ostacoli con più fiducia in me stesso. La tua presenza mi ha sempre dato un senso di sicurezza, e per questo ti sono davvero grato.
Gaetano, il "chill" fatto persona, la calma in carne e ossa. Con il tuo modo di fare pacato e rassicurante, sei riuscito a trasformare ogni momento di stress in qualcosa di più semplice da affrontare, come se tutto diventasse improvvisamente più sopportabile.
Grazie a tutti e tre per essere stati una parte fondamentale di questi anni e per aver reso ogni momento davvero speciale. Spero che questa amicizia duri nel tempo, affinché anche la serenità che ci contraddistingue non abbia mai fine.

Un sentito grazie anche agli **amici**, vicini e lontani, con cui ho condiviso momenti speciali. Che ci siamo incontrati spesso o meno, le risate, le avventure e il supporto che mi avete dato hanno reso il mio percorso più ricco e significativo.

Infine, i miei più sentiti ringraziamenti vanno a te, **Marta**, per la tua costante presenza e il tuo supporto incondizionato. In questi due anni sei stata per me tante cose: una compagna di avventure, un rifugio nei momenti difficili e un punto di riferimento quando tutto sembrava confuso. Sei stata la mia motivazione a migliorare e il mio sorriso nelle giornate più buie. Anche quando io stesso non credevo in me, tu hai sempre saputo farlo. Hai la straordinaria capacità di capirmi con un semplice sguardo o poche parole dette durante una chiamata.
Mi hai insegnato ad affrontare le sfide con coraggio e determinazione, aiutandomi a vedere la vita con occhi nuovi e pieni di speranza.
Sei un'esplosione di emozioni, ma quella che più ti rappresenta è la gioia: la gioia che porti ovunque vai e che hai saputo portare anche nella mia vita. Certo, a volte queste tue emozioni possono sembrare difficili da gestire, ma sono orgoglioso di come stai imparando a viverle pienamente, trasformandole in qualcosa di speciale che ti rende unica.

Il tuo amore, la tua forza e il tuo ascolto sono stati il mio faro nei momenti di incertezza. Ti sarò sempre grato per tutto l'amore e la felicità che porti nella mia vita ogni giorno.

Grazie a tutti voi, siete stati il mio equilibrio e la mia forza.