# Capitolo 1

# Introduzione

Negli ultimi anni, gli smart contracts hanno acquisito una crescente popolarit, attirando l'attenzione di aziende e sviluppatori grazie alle loro caratteristiche innovative, come l'esecuzione automatica, la trasparenza, l'immutabilit e la decentralizzazione. Gli smart contracts sono programmi che eseguono automaticamente azioni al verificarsi di condizioni predefinite, operando su una blockchain, un registro distribuito che mantiene una lista di record, chiamati blocchi, collegati in modo sicuro mediante crittografia.

Tutta questa attenzione viene confermata dal loro inserimento all'interno di una proposta di regolamentazione da parte dell'Unione Europea [**?**], in cui gli Smart Contracts vengono definiti come:

> "programmi informatici su registri elettronici che eseguono e regolano transazioni sulla base di condizioni prestabilite. Tali programmi possono potenzialmente fornire ai titolari e ai destinatari dei dati garanzie del rispetto delle condizioni per la condivisione dei dati."

La pi grande piattaforma di Blockchain che permette l'esecuzione degli smart contracts Ethereum. Gli smart contracts di Ethereum sono scritti in Solidity, un linguaggio di programmazione Turing-completo ad alto livello. Tuttavia, la scrittura di smart contracts sicuri rappresenta una sfida, in quanto anche piccoli errori possono portare a gravi conseguenze. Gli smart contracts sono immutabili, di conseguenza una volta pubblicati non possono essere modificati. Questa loro caratteristica, per quanto rappresenti una conferma del fatto che il contratto accettato dalle parti non possa essere modificato, pu rilevarsi il loro principale punto debole. Difatti, qualsiasi errore o vulnerabilit presente in uno smart contract non pu essere corretto dando la possibilit a soggetti malintenzionati di rubare fondi o causare altri danni.

Diventa, di conseguenza, fondamentale per gli sviluppatori disporre di tool che permettano di rilevare automaticamente le vulnerabilit presenti negli smart contracts che sviluppano. Spesso, per risolvere questo task si fa riferimento a tecniche di analisi statica, che permettono di analizzare il codice sorgente o il bytecode di uno smart contract senza

effettuarne l'esecuzione effettiva. Questo approccio consente di identificare potenziali problematiche senza la necessit di testare il codice in un ambiente reale, di contro per queste tecniche si rivelano essere molto lente e fanno troppo affidamento al riconoscimento di pattern conosciuti o si basano su dei set di regole predefinite, che possono non essere esaustive o non riconoscere nuove vulnerabilit.

Un'alternativa a queste tecniche  l'utilizzo di tecniche di Machine Learning e Deep Learning. Queste tecniche permettono di rilevare le vulnerabilit presenti negli smart contracts in maniera pi rapida e con una maggiore accuratezza rispetto alle tecniche di analisi statica. In questo campo, negli ultimi anni c' stato un notevole interesse e sono stati proposti diversi approcci che utilizzano tecniche di Machine Learning e Deep Learning per la rilevazione di vulnerabilit negli smart contracts. Questi approcci si basano sull'addestramento di modelli di Machine Learning e Deep Learning.

Questo lavoro di tesi si inserisce in questo contesto e propone un approccio basato su tecniche di Deep Learning utilizzando modelli basati sui Transformers per la rilevazione di vulnerabilit negli smart contracts. In particolare, questo lavoro ha l'obiettivo di costruire dei modelli che siano in grado di rilevare cinque classi di vulnerabilit:

- Access-Control
- Arithmetic
- Other
- Reentrancy
- Unchecked-Calls

Il contributo di questo lavoro consiste nel dimostrare l'utilizzo di modelli basati sui Transformers per la rilevazione di vulnerabilit negli smart contract. In particolare, sono stati impiegati i modelli BERT, DistilBERT e CodeBERT per la classificazione. Considerando che questi modelli presentano un limite di 512 token in input e che i contratti possono essere significativamente pi lunghi,  stato sperimentato un approccio di segmentazione del testo in sottocontratti e successiva aggregazione o concatenazione degli stessi. Questi modelli sono stati addestrati sia utilizzando il bytecode che il codice sorgente dei contratti.

Una volta ottenuti i modelli base,  stato utilizzato un modello ensamble per e ottenere un modello che effettua le predizioni a partire dalle predizioni dei migliori modelli addestrati rispettivamente sul codice sorgente e sul bytecode.

Inoltre, data la crescente diffusione di chatbot e assistenti virtuali nelle nostre vite, sia per il supporto a compiti di vario genere sia come ausilio alla programmazione,  stato testato il modello Gemini di Google per valutare la sua performance in un task di classificazione di questo tipo.

I risultati ottenuti mostrano come questi modelli siano in grado di rileva le vulnerabilit presenti negli smart contracts con delle ottime performance. In particolare, il modello CodeBERT, un modello preaddestrato su codice sorgente, ha ottenuto i risultati migliori, sia classificando il codice sorgente che classificando il codice bytecode in esadecimale.

I risultati migliori in termini di accuratezza e F1-Score sono stati ottenuti dai modelli addestrati con tre blocchi di 512 token concatenati, raggiungendo sul codice sorgente un'accuratezza dell'83% ed un F1-Score del 88%. Al contrario, i migliori risultati in termini di recall sono stati ottenuti dai modelli addestrati con tre blocchi di 512 token aggregati con la funzione max, raggiungendo una Micro Recall superiore all'85%. Il modello ensable ha visto come miglior meta-classificatore la Regressione Logistica, che ottiene una Micro Recall dell'88% e un F1-Score del 90%. Allo stesso tempo, i risultati del modello Gemini di Google non sono stati soddisfacenti, evidenziando come per task di questo genere sia necessario utilizzare modelli specificamente addestrati.

La suddivisione di questo lavoro  articolata come segue:

- il capitolo due illustra le motivazioni alla base del presente lavoro di tesi, introducendo il concetto di smart contract e sottolineando l'importanza della rilevazione delle vulnerabilit in essi presenti. Verranno inoltre descritte alcune delle principali classi di vulnerabilit riscontrabili negli smart contract. Successivamente, sar presentata una revisione della letteratura esistente in questo campo, con una panoramica dei lavori pi significativi che hanno trattato questo tema.

- il terzo capitolo espone la metodologia adottata in questo lavoro. Inizialmente, sar condotta un'analisi esplorativa del dataset, al fine di estrarne informazioni utili per la costruzione dei modelli. Successivamente, saranno introdotti i modelli utilizzati, BERT, DistilBERT e CodeBERT, con una dettagliata descrizione delle loro architetture e delle modalit di addestramento. Saranno illustrate le implementazioni e il setup dei vari esperimenti condotti. Inoltre, in questo capitolo sar presentato il modello Gemini di Google e verr spiegato come  stato applicato nel contesto di questo studio.

- il quarto capitolo riporta i risultati ottenuti nel corso del lavoro. Verranno introdotte le metriche di valutazione utilizzate e saranno presentate le performance dei vari modelli addestrati, valutate in termini di accuratezza, precisione, recall e F1 score. Infine, saranno mostrati i risultati ottenuti dai meta-classificatori e dal modello Gemini di Google.

- il quinto ed ultimo capitolo riporta le conclusioni del lavoro svolto, con una discussione delle possibili direzioni future di ricerca.