Capitolo 1

Motivazioni e Lavori Correlati

1.1 Motivazioni

1.2 Lavori Correlati

Il tema della rilevazione delle vulnerabilit all'interno degli Smart Contracts un tema che ha guadagnato notevole importanza nel tempo, anche a seguito della grande diffusione della tecnologia blockhain. Proprio per questo motivo, sono stati sviluppati e proposti diversi approcci per la rilevazione automatica di vulnerabilit all'interno degli Smart Contracts. In questo capitolo verranno presentati alcuni dei lavori pi significativi che si sono occupati di questo tema. Tra i principali approcci proposti figurano gli approcci basati su analisi statica basati su tecniche di esecuzione simbolica. L'analisi statica si basa sull'esame del codice sorgente o 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. L'esecuzione simbolica una tecnica particolarmente potente in questo contesto in quanto consente di esplorare tutte le possibili esecuzioni del programma, consentendo di individuare vulnerabilit che potrebbero emergere solo in determinate condizioni. Gli approcci basati su esecuzione simbolica cercano di risolvere queste vulnerabilit attraverso la generazione di un grafo di esecuzione simbolico, in cui le variabili sono rappresentate come simboli e le esecuzioni possibili del programma vengono esplorate simbolicamente. Ci consente di identificare percorsi di esecuzione che potrebbero condurre a condizioni di errore o vulnerabilit. Tuttavia, va notato che l'analisi statica, inclusa l'esecuzione simbolica, pu essere complessa e non sempre completa. Alcune vulnerabilit potrebbero sfuggire a questa analisi o richiedere ulteriori tecniche di verifica. Pertanto, consigliabile combinare l'analisi statica con altre metodologie, come l'analisi dinamica e i test formali, per garantire una copertura completa nella rilevazione di vulnerabilit negli smart contract. Tra i principali strumenti che utilizzano questo tipo di analisi ci sono Maian [**?**, **?**], Oyente [**?**, **?**], Mythril

[?], Manticore [?] e altri. Un'altro tipo di approcci ad analisi statica sono i tools basati su regole. Questi strumenti usano un set di regole predefinite e pattern per identificare delle potenziali nvulnerabilit nel codice sorgente. Questi tool analizzano il codice sorgente e segnalano tutte le istanze dove il codice viola delle regole predefinite. Le regole sono tipicamente basate su delle vulnerabilit note e delle best practice di programmazione, come ad esempio evitare dei buffer overflow, usare algoritmi di cifratura sicuri e validare propriamente l'input degli utenti. La limitazione principale di questi strumenti che i risultati che producono sono limitati al set di regole che stato implementato, quindi non riescono a riconoscere delle nuove vulnerabilit o vulnerabilit non scoperte precedentemente. Inoltre, un'altra grande limitazione di questi strumenti il fatto che possano produrre un alto numero di falsi positivi, cio di situazioni in cui il codice viene segnalato come codice vulnerabile ma in realt codice perfettamente sano. Tra i principali strumenti che utilizzano questo tipo di analisi ci sono Slither [?], Securify [?], SmartCheck [?] e altri. Un'altra categoria di strumenti per la rilevazione di vulnerabilit negli smart contract sono gli approcci basati su tecniche di Machine Learning e Deep Learning, tra le quali anche il lavoro di questa tesi va ad inserirsi. Un approccio basato sulla trasformazione degli opcode dei contratti e la sua relativa analisi con dei modelli di Machine Learning tradizionale stato offerto da Wang et al. [?] i quali hanno raggiunto risultati eccellenti, con risultati in termini di F1 Score superiori al 95% in quasi tutte le classi prese in analisi con il modello XGBoost che risultato il miglior modello. Un altro lavoro che sfrutta tecniche di Machine Learning pi tradizionali il lavoro di Mezina e Ometov che hanno utilizzato classificatori come RandomForest, LogisticRegressio, KNN, SVM in un approccio dapprima binario (valutare se il contratto abbia o meno delle vulnerabilit) e poi multiclasse (valutare quale tipo di vulnerabilit il contratto abbia) [?]. I risultati in questo caso hanno mostrato come il modello SVM sia quello che ottiene i migliori risultati in termini di accuratezza.

Spostandoci su lavori che utilizzano tecniche di Deep Learning importante citare un'altro lavoro effettuato sullo stesso dataset su cui basato questo lavoro di tesi. Questo dataset stato infatti raccolto e pubblicato da un gruppo di ricercatori dell'Universit di Bologna che ha effettuato un primo studio utilizzando un approccio basato su reti neurali convoluzionali [?]. L'approccio in questo caso stato quello di classificare le vulnerabilit in un'impostazione multilabel del problema (in cui la label da predire un array di elementi, quindi in cui il contratto pu appartenere contemporaneamente a pi classi) itilizzando delle reti neurali convoluzionali per la rilevazione delle vulnerabilit trasformando in codice Bytecode espresso in esadecimale dei contratti in delle immagini RGB. Questo lavoro ha come risultato principale la dimostrazione che utilizzando delle reti neurali convoluzionali possibile rilevare le vulnerabilit presenti negli SmartContracts con delle buone performance, i migliori risultati si attestano con un MicroF1 score del 0.83% e mostrano come i migliori risultati siano dati da delle resi Resnet con delle convoluzioni unidimensionali. Successivamente, gli stessi autori hanno pubblicato una seconda analisi effettuata sul dataset utilizzando nuovi classificatori come CodeNet, SvinV2-T e

Inception, mostrando come i migliori risultati continuino ad essere quelli forniti da reti convoluzionali unidimensionali [**?**]. Altri lavori che utilizzano un approccio basato su tecniche di Deep Learning il lavoro proposto da Huang [**?**] che utilizza anch'egli delle reti neurali convoluzionali per la rilevazione delle vulnerabilit. I modelli utilizzati sono modelli molto noti come Alexnet, GoogleNet e Inception v3, i risultati migliori in questo caso si attestano sul 75%. Un importante lavoro offerto da Deng et Al. [**?**] ha proposto un approccio basato sulla fusione di feature multimodali, analizzando contemporaneamente codice sorgente, bytecod e grafi di controllo del flusso. Per ognuna di queste tre feature stato trainato un semplice classificatore con una rete neurale feedforward e sono poi state combinate le predizioni di questi tre classificatori in un classificatore finale utilizzando un approccio di ensamble learning detto stacking. I risultati ottenuti mostrano come l'approccio proposto abbia ottenuto dei risultati migliori rispetto ad un approccio in cui si utilizzava solo una delle tre feature.