Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Machine Learning Vulneralibities Detection in SmartContracts .....

Relatore:
Chiar.mo Prof.
Stefano Ferretti

Presentata da:
Marco Benito Tomasone

Correlatore:
Dott. Stefano Pio Zingaro
Dott. Saverio Giallorenzo

Sessione I
Anno Accademico 2023/2024

Alle nonne

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nel mondo delle tecnologie blockchain, gli smart contracts hanno guadagnato popolarità grazie alla loro abilità di automatizzare e far rispettare gli accordi tra due soggetti senza la necessità di un intermediario. Negli ultimi anni una delle tecnologie che sono spopolate e sono arrivate sulla bocca di tutti sono sicuramente la Blockchain e gli SmartContracts. Questi ultimi sono dei contratti digitali che permettono di eseguire delle operazioni in modo automatico e trasparente. Questi contratti sono scritti in un linguaggio di programmazione e vengono eseguiti su una macchina virtuale. Una delle principali caratteristiche peculiari degli SmartContracts è sicuramente la loro immutabilità, difatti una volta essere stati deployati sulla blockchain questi non possono più essere modificati. Uno dei problemi principali degli SmartContracts è la sicurezza. Infatti, essendo dei contratti che vengono eseguiti in modo automatico, è possibile che ci siano delle vulnerabilità che possono essere sfruttate da malintenzionati. In questo lavoro di tesi verrà presentato un metodo per rilevare le vulnerabilità presenti negli SmartContracts.

Capitolo 2

Related Works

In prima battuta questo lavoro è stato possibile grazie ad un lavoro precedente che ha permesso di creare un dataset di SmartContracts [4]. Questo dataset è stato creato da un gruppo di ricercatori dell'Università di Bologna. Questo primo lavoro su questo datataset ha principalmente impiegato delle tecniche di Deep Learning utilizzando delle reti neurali convoluzionali per la rilevazione delle vulnerabilità trasformando in codice Bytecode dei contratti in delle immagini RGB. Questo lavoro ha come risultato principale la dimostrazione che utilizzando delle reti neurali convoluzionali è possibile rilevare le vulnerabilità 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 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 [**?**].

Capitolo 3

Dataset

Il dataset [3] utilizzato in questo progetto è un dataset disponibile pubblicamente sulla piattaforma HuggingFace una delle più importanti piattaforme per il Natural Language Processing. HF è un'infrastruttura open-source che fornisce accesso a una vasta gamma di modelli di deep learning pre-addestrati, tra cui alcuni dei più avanzati nel campo del NLP. Questo dataset contiene 106.474 smart contracts scritti in linguaggio Solidity un linguaggio di programmazione Touring completo divenuto lo standard per la scrittura di smart contracts su Ethereum. Ogni elemento nel dataset è composto da quattro elementi:

- **Address**: l'indirizzo del contratto
- **SourceCode**: il codice sorgente del contratto
- **ByteCode**: il codice bytecode del contratto
- **Slither**: il risultato dell'analisi statica del contratto con Slither, è un array che contiene valori che vanno da 1 a 5.

3.0.1 Vulnerabilità

In questo lavoro sono state prese in considerazione cinque classi di vulnerabilità diverse:

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

Access-Control

Arithmetic

Le vulnerabilità di tipo aritmetico [1] sono vulnerabilità che vengono generate come risultato di operazioni matematiche. Una delle vulnerabilità più significative all'interno di questa classe è rappresentata dagli underflow/overflow, un problema molto comune nei linguaggi di programmazione. Incrementi di valore di una variabile oltre il valore massimo rappresentabile o decrementi al di sotto del valore minimo rappresentabile (detti *wrap around*) possono geneare comportamenti indesiderati e risultati errati.

Other

Reentrancy

La Reentrancy è una classe di vulnerabilità presente negli SmartContracts che permette ad un malintenzionato di rientrare nel contratto in modo inaspettato durante l'esecuzione della funzione originale. Questa vulnerabilità può essere utilizzata per rubare fondi e rappresenta la vulnerabilità più impattante dal punto di vista di perdita di fondi a seguito di attacchi. Il caso più famoso di questo attacco che lo ha anche reso noto è il caso di The DAO, un contratto che ha subito un attacco di reentrancy che ha portato alla perdita circa sessanta milioni di dollari in Ether, circa il 14% di tutti gli Ether in circolazione all'epoca. Nonostante dal 2016 ad oggi siano stati fatti numerosi progressi nelle tecnologie e nelle misure di sicurezza questa vulnerabilità rimane comunque una delle minacce più pericolose per gli SmartContracts, poichè negli anni questo tipo di attacchi si è ripresentato notevole frequenza [2]. Un attacco di reentrancy può essere classificato in tre classi differenti:

- **Mono-Function**: la funzione vulnerabile è la stesa che viene chiamata più volte dall'attaccante, prima del completamento delle sue invocazioni precedenti. Questo è il caso più semplice di attacco reentrancy e di conseguenza il più facile da individuare.

- **Cross-Function**: questo caso è molto simile al caso di mono-function Reentrancy, ma in questo caso la funzione che viene chiamata dall'attaccante non è la stessa che fa la chiamata esterna. Questo tipo di attacco è possibile solo quando una funzione vulnerabile condivide il suo stato con un'altra funzione, risultando in un una situazione fortemente vantaggiosa per l'attaccante.

- **Cross-Contract**: questo tipo di attacco prende piede quando lo stato di un contratto è invocato in un altro contratto prima che viene correttamente aggiornato. Avviene solitamente quando più contratti condividono una variabile di stato comune e uno di loro la aggiorna in modo non sicuro.

Mostreremo adesso alcuni esempi di contratti vulnerabili a questo tipo di attacco.

```
// UNSECURE
function withdraw() external {
    uint256 amount = balances[msg.sender];
    (bool success,) = msg.sender.call{value: balances[msg.sender
       ]}("");
    require(success);
    balances[msg.sender] = 0;
}
```

In questo caso, il balance dell'utente viene aggiornato solo dopo che la chiamata esterna è stata completata. Questo permette all'attaccante di chiamare la funzione withdraw più volte prima che il balance venga settato a zero, permettendo all'attaccante di rubare fondi allo smart contract. Una versione più complessa dello stesso processo è il caso cross function, di cui mostriamo un esempio:

```
    // UNSECURE
function transfer(address to, uint amount) external {
  if (balances[msg.sender] >= amount) {
    balances[to] += amount;
    balances[msg.sender] -= amount;
  }
}

function withdraw() external {
  uint256 amount = balances[msg.sender];
  (bool success,) = msg.sender.call{value: balances[msg.sender]}(
     "");
  require(success);
  balances[msg.sender] = 0;
}
```

In questo esempio, l'attaccante può effettuare un attacco di tipo reentrancy avendo una avendo una funzione che chiama `transfer()` per trasferire fondi spesi prima che il bilancio sia settato a zero dalla funzione `withdraw()`. Un nuovo tipo di attacchi sono gli attacchi Read-only Reentrancy, in

Unchecked-Calls

In solidity si possono usare delle chiamate a funzione low level come 'address.call()'

3.1 Exploratory Data Analysis

Prima della costruzione dei modelli è stata affrontata una fase di analisi esplorativa dei dati. Questa fase è stata svolta per comprendere meglio la struttura del dataset e dei contratti che si andavano a classificare, per individuare eventuali problemi. Il dataset è diviso in tre sottoinsiemi: training, validation e test set. Il dataset di training è composto da 79.641 contratti, il dataset di validazione da 10.861 contratti e il dataset di test da 15.972 contratti.

Capitolo 4

Metodologia

4.1 BERT

Il modello BERT (Bidirectional Encoder Representations from Transformers) rappresenta un pilastro fondamentale nel campo del Natural Language Processing (NLP), grazie alla sua capacità di comprensione del contesto delle parole all'interno di una frase o di un testo più ampio. Questo modello, sviluppato da Google, si basa sull'architettura dei transformer, una classe di modelli neurali che ha dimostrato notevole successo nell'analisi del linguaggio naturale.

BERT si distingue per la sua capacità bidirezionale di elaborare il contesto linguistico. A differenza dei modelli NLP precedenti, che processavano il testo in modo sequenziale, interpretando le parole una dopo l'altra, BERT considera sia il contesto precedente sia quello successivo di ciascuna parola all'interno di una frase. Questo approccio bidirezionale consente a BERT di catturare relazioni semantiche più complesse e di fornire una rappresentazione più accurata del significato del testo.

Il funzionamento di BERT può essere compreso attraverso due fasi principali: l'addestramento e l'utilizzo.

Durante la fase di addestramento, BERT viene esposto a enormi quantità di testo, proveniente da varie fonti e domini. Utilizzando un processo noto come "pre-addestramento", il modello apprende i modelli linguistici e il contesto delle parole. Questo pre-addestramento coinvolge due compiti principali: la predizione di parole mascherate e la predizione della successione di frasi. Nel primo compito, BERT impara a prevedere le parole mancanti in una frase fornita, mentre nel secondo compito, il modello apprende a determinare se due frasi sono consecutive in un testo o sono state estratte casualmente da testi diversi.

Una volta completata la fase di addestramento, BERT può essere utilizzato per una vasta gamma di compiti NLP senza la necessità di ulteriori addestramenti specifici. Durante l'utilizzo, il modello riceve in input una sequenza di token, che possono essere

parole, frammenti di testo o segmenti di frasi. Ogni token viene rappresentato come un vettore di caratteristiche, derivato dal contesto bidirezionale fornito da BERT durante l'addestramento. Queste rappresentazioni vettoriali possono essere utilizzate per svariati compiti, come classificazione di testo, analisi del sentiment, risposta alle domande, traduzione automatica e molto altro ancora.

In sintesi, il modello BERT ha rivoluzionato il campo del NLP introducendo una comprensione più approfondita e contestualizzata del linguaggio naturale. La sua capacità di catturare il contesto bidirezionale delle parole ha portato a miglioramenti significativi nelle prestazioni dei sistemi NLP su una vasta gamma di compiti e applicazioni. BERT rimane un pilastro fondamentale nell'ambito della comprensione automatica del linguaggio umano, consentendo a macchine e sistemi di interagire e comprendere il linguaggio umano in modo più naturale e preciso.

Bibliografia

[1] Zulfiqar Ali Khan and Akbar Siami Namin. Ethereum smart contracts: Vulnerabilities and their classifications. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1–10, 2020.

[2] pcaversaccio. A historical collection of reentrancy attacks. https://github.com/pcaversaccio/reentrancy-attacks, 2023.

[3] Martina Rossini. Slither audited smart contracts dataset, 2022.

[4] Martina Rossini, Mirco Zichichi, and Stefano Ferretti. On the use of deep neural networks for security vulnerabilities detection in smart contracts. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 74–79, 2023.