Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Stimare i consumi di gas nell'esecuzione di Smart Contract in Ethereum

Relatore:
Chiar.mo Prof.
Ugo Dal Lago

Presentata da:
Melania Ghelli

Sessione II
Anno Accademico 2018-2019

Questa è la Dedica*:*
ognuno può scrivere quello che vuole,
anche nulla ...

Indice

Introduzione

Le innovazioni tecnologiche introdotte negli ultimi decenni hanno rivoluzionato la nostra società. Il risultato che ne deriva è che numerosi settori stanno cambiando, muovendosi verso una realtà sempre più digitale. Se da una parte questo ha costituito un progresso, dall'altra ha creato nuove opportunità per i cybercriminali.
Oggi il principale obiettivo della sicurezza informatica è proprio quello di trovare soluzioni adattabili alle nuove infrastrutture, come i sistemi IoT che si stanno diffondendo sempre di più. Con l'impiego di queste nuove tecnologie nell'industria i punti di accesso alla rete aziendale sono aumentati, moltiplicando la tipologia ed il numero di minacce. In questo contesto il rischio che si corre è maggiore, poichè legato alla violazione di dati sensibili o addirittura alla compromissione dei processi di produzione.
La blockchain nasce in questo contesto, riscuotendo un grande successo grazie al potenziale innovativo che porta con sé. Questa nuova tecnologia permette l'esecuzione di programmi in modo distribuito e sicuro, senza la necessitá di un ente centrale che faccia da garante. Il paradigma trova applicazioni nei settori piú disparati, offrendo innovazione grazie alla possibilitá di fare a meno di banche o istituzioni pubbliche.
Ma in cosa consiste dal punto di vista informatico? Blockchain, catena di blocchi. Registro distibuito, organizzato in blocchi legati tra loro. Ad interagire con essa sono i miner, coloro che fisicamente realizzano le così dette transazioni. Qual è il ruolo del miner nello specifico??

Grazie alla crittografia (sicurezza informatica comprende una serie di cose, tra cui la crittografia) è stato possibile implementare delle monete virtuali. Da questi studi si è arrivati poi a pensare al bitcoin, che è stato il primo esperimento condotto con successo.

Ciò che indeboliva le crittovalute era il fatto di essere "centralizzate", cioè di passare per un ente che ne garantisse l'uso (una sorta di banca).
La blockchain nasce come necessità di superare quest'ostacolo, perché finalmente introduce la possibilità di farne a meno. Il primo esempio storico di moneta digitale si ha con il Bitcoin. Già da lui si parla di assenza di autorità centrale e rete distibuita - usando algoritmi proof-of-work. Questi permettono di raggiungere un consenso distribuito attraverso tutta la rete. E' il concetto chiave x capire come riusciamo ad evitare di affidarci ad un singolo ente es. Banca.
Tra i sistemi nati grazie alla blockchain troviamo Ethereum, una piattaforma che mette a disposizione un linguaggio di programmazione di alto livello. Questo linguaggio puó essere utilizzato dagli utenti per implementare dei programmi, i così detti smart contract. In che cosa differisce dal Bitcoin?

Gli smart contract possono essere eseguiti sulla rete di Ethereum solo al fronte di un pagamento anticipato. Per ragioni di sicurezza a ciascuna istruzione di basso livello è associato un costo monetario. Dunque eseguire un programma costerá tanto quante sono le istruzioni che lo compongono. Il costo di ciascuna istruzione è espresso in termini di gas, una sorta di carburante che viene pagato in ether, la crittovaluta di Ethereum. Dal momento che il gas viene pagato in anticipo, potrebbe accadere che l'esecuzione di un programma ecceda la quantitá messa a disposizione. In questi casi la computazione non giunge al termine, risultando nella perdita delle risorse investite dall'utente. Oltre a questo comportamento indesiderato l'esaurimento del gas disponibile puó avere conseguenze pericolose. Un programma che non gestisce correttamente queste situazioni viene etichettato come vulnerabile. La conseguenza piú diretta è il blocco del contratto, che puó essere anche permanente. La pericolositá peró risiede nel fatto che questo tipo di programmi diventano un bersaglio facile per attacchi malevoli. Vedremo come queste vulnerabilitá possono essere sfruttate per ottenere comportamenti dannosi per la rete.
Dato il valore monetario associato agli smart contract il rischio che si corre in caso di attacchi informatici è una perdita di denaro. Per questo motivo è necessario individuare possibili criticitá nel codice prima della sua esecuzione. In questo contesto l'analisi statica dei programmi costituisce un potente strumento di prevenzione.

All'interno di questo elaborato ci concentreremo solo sulle tecniche di analisi dei consumi di gas. Poter conoscere a priori quest'informazione permetterebbe non solo un investimento adeguato da parte degli utenti, ma anche uno strumento di prevenzione da possibili attacchi.
Attualmente non esistono strumenti in grado di calcolare con precisione la quantitá di gas richiesto durante una computazione. Cercheremo di capirne le ragioni ma soprattutto di individuare dei margini di miglioramento.

Capitolo 1

Background

Lo scopo di questo capitolo è quello di fornire una panoramica dei concetti chiave intorno ai quali si sviluppa l'elaborato.

1.1 Blockchain

Il termine Blockchain - in italiano "catena di blocchi" - identifica un registro distribuito e sicuro. In questo senso si può pensare alla blockchain come ad una struttura di dati simile ad una lista crescente, dove le informazioni sono raggruppate in blocchi collegati fra loro.
Ciascun blocco codifica una sequenza di transazioni individuale, e viene concatenato a quello precedente seguendo un ordine cronologico. La concatenazione è irreversibile: ciascun nuovo blocco contiene la firma digitale di quello precedente. In questo modo, modificare un blocco implicherebbe l'invalidazione di tutta la catena successiva.
La peculiarità di questa struttura risiede nel fatto che sia condivisa: ogni nodo che compone la rete mantiene una copia del registro aggiornata. Per poter aggiungere un blocco è dunque necessario validare l'intera catena, ed ottenere un consenso da parte degli altri nodi della rete. Una volta ottenuto, il nuovo blocco viene trasmesso agli altri componenti in modo tale da aggiornare lo stato della blockchain.
Il processo di validazione dei nuovi blocchi viene realizzato dai miner. Il loro compito è quello di verificare le transazioni proposte e fare in modo che il nuovo blocco venga

linkato alla blockchain. Per fare questo i miner sono chiamati a risolvere un algoritmo proof-of-work, un puzzle crittografico che richiede un significativo costo computazionale per essere risolto.
Questo sistema permette di raggiungere il consenso senza la necessità di un'autorità centrale che faccia da garante. É il concetto chiave delle tecnologie basate su blockchain: la possibilità di implementare servizi sicuri senza appoggiarsi a banche, istituzioni pubbliche, ecc.

Questa nuova tecnologia può essere integrata in diverse aree [7], sebbene ad oggi il suo uso più conosciuto sia quello nei sistemi di pagamento che impiegano crittovalute. Il dato non è poi così sorprendente: la prima blockchain nasce grazie a Satoshi Nakamoto assieme al Bitcoin [8]. In questo senso il Bitcoin è una piattaforma di pagamenti, dove la catena di blocchi funge da storico di tutte le transazioni avvenute: una sorta di conto corrente condiviso.

1.2 Ethereum, gli Smart Contract e la EVM

All'interno di quest'elaborato verrà presa in considerazione solo il network Ethereum, una piattaforma decentralizzata basata su una blockchain, che come Bitcoin possiede una propria valuta: l'*ether*.
Diversamente da quanto vale per le altre crittovalute, Ethereum non è solo un network per lo scambio di moneta, ma un framework che permette l'esecuzione di programmi. Tali programmi prendono il nome di *smart contract*, cioè "contratti intelligenti". Sebbene il nome possa suggerire una funzione ben precisa, questi programmi sono usati per computazioni general-purpose, permettendo quindi di realizzare un vasto numero di operazioni.
Gli smart contract sono scritti in linguaggi ad alto livello; fra i vari (Serpent, Viper e LLL) quello più diffuso ad oggi è Solidity [3]. Tale linguaggio object-oriented è pensato solo per lo sviluppo di smart contract che, per poter girare nella rete, vengono poi tradotti in bytecode. Ciascun nodo di Ethereum infatti esegue localmente la Ethereum Virtual

Machine, anche detta EVM, una macchina a stack in grado di eseguire un linguaggio di basso livello, ossia bytecode. Questo linguaggio è non tipato, e composto da un piccolo insieme di istruzioni.

1.3 Il ruolo del gas

Per *gas* si intende l'unità di misura dello sforzo computazionale richiesto dalla EVM per eseguire ciascuna istruzione. Diremo quindi che eseguire uno smart contract costa una certa quantità di gas.
Nello specifico ciascuna istruzione di basso livello ha associato un costo fisso in gas. Per calcolare quindi il consumo totale di un programma Solidity è necessario comprendere in quali istruzioni di basso livello verrà tradotto.
I costi di alcune delle istruzioni EVM [9] sono riportati nella tabella 1.1.

Il gas viene pagato in ether dagli utenti che intendono far eseguire una *transazione*. Con transazione si intende l'azione di creare uno smart contract o di chiamarne delle funzioni e di pagare un miner per far eseguire la transazione stessa.
Per fare in modo che la sua transazione venga scelta, un utente stabilisce la quantità di gas che è disposto a pagare per farla portare a termine. Il miner poi, in base a questo parametro, sceglie quali transazioni eseguire. É importante fornire un'adeguata somma: i miner, al termine della transazione, possono beneficiare dell'ether destinato all'acquisto di gas che è rimasto inutilizzato. Questo significa che gli smart contract con più probabilità di essere scelti sono quelli degli utenti disposti a pagare di più.
La scelta adottata da Ethereum di far pagare i propri utenti non è triviale. Prima di tutto impedisce loro di sovraccaricare i miner di lavoro sfruttando il potere computazionale della rete. Inoltre scoraggia gli utenti a impiegare troppa memoria, una risorsa preziosa nelle tecnologie basate su blockchain. Infine limita il numero di computazioni eseguite dalla stessa transazione. Questo difende il network intero da attacchi malevoli come i DDoS: la finitezza delle transazioni fa sì che non si possa, ad esempio, far ciclare un programma infinitamente. Per poter disabilitare la rete anche solo per pochi minuti gli hacker dovrebbero pagare delle ingenti somme.

Istruzione	Costo	Descrizione
JUMPDEST	1	Indica la destinazione di un'istruzione JUMP
POP	2	Rimuove un elemento dallo stack
PUSHn	3	Inserisce un elemento di n byte nello stack
ADD, SUB	3	Operatori aritmetici
AND, OR, NOT, XOR, ISZERO, BYTE	3	Operatori logici
MUL, DIV	5	Operatori aritmetici
JUMP	8	Salto semplice senza condizione
JUMP1	10	Salto condizionale
CALL	700	Chiama una transazione
CALLVALUE	9000	Pagato per un argomento diverso da 0 dell'istruzione CALL
SSTORE	20000	Salva una parola in memoria. Si paga quando il valore precedente è uguale a 0

Tabella 1.1: Costi espressi in gas di alcune istruzioni della EVM

Dal momento che il gas viene pagato anticipatamente, può succedere che durante la sua esecuzione un programma ecceda la quantità che ha a disposizione. Questo comportamento è indesiderato, in quanto comporta spiacevoli conseguenze. La più immediata è il blocco della transazione: la computazione non giunge a termine, l'utente non ottiene il risultato desiderato e l'ether pagato per il gas va perso.
La conseguenza indiretta invece è il possibile blocco permanente dello smart contract. Quando durante l'esecuzione di una transazione un'istruzione richiede una quantità di gas superiore a quella disponibile, la EVM solleva un'eccezione di tipo *out-of-gas* e interrompe la transazione. Qualora lo smart contract non preveda una gestione di questo tipo di eccezione resterà bloccato per sempre.

Capitolo 2

Analisi statica

2.1 Tecniche in informatica

Analisi statica vs. analisi dinamica

Per analisi statica si intende l'analisi dei programmi fatta prima della loro esecuzione. L'analisi può essere fatta sia sul codice sorgente, che sul codice oggetto, vale a dire il prodotto della compilazione.
Le tecniche di analisi possono essere suddivise in due tipologie in base al loro obiettivo. La prima categoria comprende i tool di analisi volti a localizzare bug nel codice. La seconda identifica invece un gruppo di software con una forte base logica, che utilizzano tecniche matematiche per la verifica di specifiche prorietà del programma.

Per analisi dinamica si intende quella fatta durante l'esecuzione dei programmi su un processore reale o virtuale. Il software testing rientra in questa categoria.

2.2 Analisi statica di smart contract

L'impiego delle tecniche di analisi statica per la verifica degli smart contract non è molto diffuso. Principalmente perchè data la dimensione limitata di questi programmi non si ritiene necessario l'impiego dell'analisi. In parte questo è dovuto anche alla difficile rappresentazione del bytecode EVM. Decompilare le istruzioni di basso livello al fine di

ottenere una rappresentazione migliore che funga da base per una buona analisi richiede un notevole sforzo.

2.3 Tool per l'analisi

Durante questo lavoro è stato preso in considerazione un certo numero di software che implementano tecniche di analisi statica orientata alla verifica degli smart contract. Di seguito ne faremo un elenco.

EtherTrust [5] questo framework offre la possibilità di analizzare i programmi scritti al fine di verificarne le proprietà di sicurezza. Per fare questo EtherTrust traduce il bytecode in clausole Horn. I contributi di questo framework sono: la realizzazione di una semantica per EVM, la modellizzazione delle proprietà di sicurezza degli smart contract e l'analisi statica applicata al bytecode;

MadMax [4] attraverso la combinazione di più tecniche di analisi statica, questo software propone un'analisi di smart contract volta ad individuare bug legati all'esaurimento del gas disponibile;

KEVM [6] produce una semantica per la EVM. Questa modellizzazione puà essere utilizzata come base di partenza per implementare un'analisi esaustiva del codice. Gli stessi autori del programma fanno riferimento ad una possibile applicazione di KEVM volta a stimare i consumi di gas dei programmi;

EthIR [1] è un framework di analisi del bytecode di EVM. A partire dalle istruzioni di basso livello, EthIR produce una rappresentazione RB (*Ruled Based*). Tale modellizzazione può essere utilizzata per desumere proprietà del bytecode, applicando delle ulteriori tecniche di analisi statica;

GASTAP [2] è la prima piattaforma sviluppata in grado di analizzare smart contract al fine di dare un upper bound ai consumi di gas dello stesso. Questo software è ancora in via di sviluppo, perciò presenta ancora delle limitazioni. Tuttavia si distingue per la precisione nella stima dei bound, riuscendo a fornire un analisi più precisa rispetto ad altri software che implementano le stesse funzionalità.

Capitolo 3

Risultati sperimentali

Lo scopo di questo capitolo è quello di fornire una panoramica sugli esperimenti condotti.

Spiego che cosa ho fatto.

Dato un set di smart contract molto semplici, sono state condotte delle analisi utilizzando i software disponibili precedentemente elencati. A partire dai risultati ottenuti si è individuato quali di questi programmi fornissero un bound esplicito ai consumi di gas associati all'esecuzione dei programmi.

Conclusioni

Bibliografia

[1] Elvira Albert, Pablo Gordillo, Benjamin Livshits, Albert Rubio, and Ilya Sergey. Ethir: A framework for high-level analysis of ethereum bytecode. In *International Symposium on Automated Technology for Verification and Analysis*, pages 513–520. Springer, 2018.

[2] Elvira Albert, Pablo Gordillo, Albert Rubio, and Ilya Sergey. GASTAP: A gas analyzer for smart contracts. *CoRR*, abs/1811.10403, 2018.

[3] ethereum. The solidity contract-oriented programming language, 2019.

[4] Neville Grech, Michael Kong, Anton Jurisevic, Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis. Madmax: Surviving out-of-gas conditions in ethereum smart contracts. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA):116, 2018.

[5] Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind. Foundations and tools for the static analysis of ethereum smart contracts. In *International Conference on Computer Aided Verification*, pages 51–78. Springer, 2018.

[6] Everett Hildenbrandt, Manasvi Saxena, Xiaoran Zhu, Nishant Rodrigues, Philip Daian, Dwight Guth, and Grigore Rosu. Kevm: A complete semantics of the ethereum virtual machine. Technical report, 2017.

[7] Marco Iansiti and Karim R. Lakhani. The truth about blockchain. *Harvard Business Review*, 2017.

[8] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[9] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.