Approcci automatizzati per l’analisi di Smart contract

ALBERTO AMEGLIO – MATRICOLA 950144

30/06/2023

1 Shielder S.r.l.

La Schielder S.r.l è un'azienda che opera nel campo della consulenza informatica offrendo servizi di analisi in campi come la sicurezza applicativa, di rete e dell'IoT. Ciò che ha distinto la mia esperienza in questa azienda è stata l'atmosfera giovanile e stimolante che caratterizza l'ambiente di lavoro. Questa azienda ha creato un ecosistema in cui l'innovazione e la crescita personale sono incoraggiate e valorizzate.

2 Svolgimento del tirocinio

Il periodo di tirocinio presso la Schielder S.r.l. mi ha permesso di interagire e stabilire relazioni con la maggior parte dei dipendenti. Gli incontri sono avvenuti sia durante l'orario di lavoro, in smart working, o presso la sede dell’azienda a Pinerolo, oppure in occasioni di speciali meet up, organizzati per favorire la reciproca conoscenza tra i dipendenti.

Durante il mio percorso di tirocinio, ho avuto come tutor aziendale, il Dott. Davide Silvetti, che opera nel campo della vulnerability research e penetration testing, con oltre 5 anni di esperienza in Schielder, mentre, per il coordinamento didattico, ho potuto fare affidamento sul Prof. Marco Anisetti.

Nel corso del tirocinio, ho approfondito le conoscenze nell'ambito del Web3, esplorando i nuovi protocolli offerti dai sistemi decentralizzati. L’obiettivo principale è stato approfondire la comprensione degli smart contract di Ethereum ed il linguaggio di programmazione Solidity, che rappresentano l'elemento centrale nello sviluppo di questa nuova frontiera del Web, costituendo la base per numerose applicazioni decentralizzate e anche per la DeFi.

Il mio percorso è stato caratterizzato da un alto grado di autonomia e da un'intensa attività di studio e ricerca. Con l’obiettivo di arricchire la mia conoscenza, ho consultato libri, blog e CTF (Capture The Flag), che non hanno avuto l’obiettivo di una competizione, ma hanno permesso un ambiente educativo stimolante in cui ho potuto esplorare le metodologie legate alla sicurezza informatica negli smart contract.

Ho intrapreso un approccio nel campo del reversing degli smart contract, che implica un'analisi dettagliata dell'opcode per esaminare i contratti da un’altra prospettiva. Questa metodologia ha notevolmente ampliato la mia comprensione dei meccanismi interni dell'ambiente di esecuzione EVM.

Ho inoltre potuto esplorare i tool open source di maggiore rilevanza nel settore per l’analisi automatizzata ed il fuzzing, con un focus sull’utilizzo di Echidna, un potente strumento di analisi e testing automatizzato, scritto in Haskell e sviluppato da Trail of Bits.

3 Obiettivi del lavoro

Per la realizzazione della tesi, ho sviluppato un sistema di analisi automatica delle vulnerabilità dei contratti in Solidity, che è in grado di condurre verifiche di sicurezza sul codice sorgente. L'obiettivo principale del sistema è identificare le vulnerabilità più comuni, mantenendo una discreta soglia di falsi positivi. Nel dettaglio, il software può essere classificato come un property-based tester che implementa libreria Hypothesis. Questa ci consente di descrivere come appaiono gli input invece di specificare cosa sono. In questo modo vengono generati molti casi di test con input corrispondenti a quella descrizione. Il framework si arresta quando incontra un test case fallito e genera un errore specificando l’input che ha bloccato il flusso.

Sviluppare questo fuzzer avrebbe dovuto ampliare la gamma dei tool, ad oggi presenti in rete, che rappresentano uno dei più grandi alleati agli Unit test. Questo obiettivo è stato ribadito nelle ultime evidenze presentate [1] dal team di

auditing di Trail of Bits, che ha confermato che non sono riscontrabili correlazioni statisticamente significative tra la qualità degli Unit test e la quantità di vulnerabilità gravi riscontrate nei vari sistemi.

4 Panoramica sul lavoro svolto

Nel corso di questo elaborato, ho eseguito una panoramica completa dei passi chiave che hanno condotto all'evoluzione di Ethereum. Il percorso è iniziato da Bitcoin, come pietra miliare nel panorama delle criptovalute e delle blockchain, e da una breve comprensione dei concetti che compongono il Web3.

Una parte significativa è stata dedicata all'analisi delle vulnerabilità e alla ricerca delle stesse. Questa parte comprende una descrizione dettagliata dei tipi di vulnerabilità più comuni, illustrando come queste minacce possano essere sfruttate dagli attaccanti. Nel corso di questa ricerca, verranno esplorati una serie di strumenti e tecniche ampiamente riconosciute e utilizzate per l'analisi.

5 Tecnologie decentralizzate

Il termine Web3, venne coniato nel 2014 da Gavin Wood, co-fondatore di Ethereum. E' utilizzato per indicare una nuova fase di internet in cui i beni, l'identità digitale e la decentralizzazione dei dati, rappresentano una priorità. Il Web, per come lo conosciamo oggi, richiede troppa fiducia in quanto gran parte delle applicazioni Web conosciute e usate dagli utenti sono gestite e mantenute da poche aziende private, che non sempre hanno un vantaggio ad agire nell'interesse pubblico.

Grande attenzione verso i sistemi decentralizzati è iniziata con Bitcoin, tecnologia preceduta da un periodo di ricerca e sviluppo nella crittografia e nelle reti peer-to-peer. Il concetto di denaro digitale è stato per la prima volta formulato da David Chaum negli anni '80, ma è stato Bitcoin, introdotto nel 2009 da Satoshi Nakamoto, attraverso la sua prima transazione con Hal Finney, a rendere questa visione una realtà. Bitcoin è una moneta digitale memorizzata su una blockchain, definibile come il registro di tutte le transazioni che vengono effettuate con questa moneta in un sistema pubblico, immutabile e distribuito. Il whitepaper di Bitcoin [2] descrive la visione e il funzionamento, partendo da molteplici scoperte che lo hanno preceduto:

- "How to Time Stamp a Digital Document" [3] che evidenzia come sia possibile utilizzare algoritmi di hashing crittografico per disincentivare comportamenti scorretti nelle comunicazioni tra più parti.
- "The Weak Byzantine Generals Problem" [4] che descrive un metodo per raggiungere un accordo tra un insieme di processi, nonostante la possibilità che alcuni siano scorretti.
- "A Certified Digital Signature" [5] in cui viene descritta una struttura di autenticazione ad albero
- "Pricing via Processing or Combatting Junk Mail" [6] un sistema per disincentivare lo spam nell'email, sistema che ha trovato utilizzo del Proof of Working.

Bitcoin è limitato nelle sue funzionalità da alcune scelte di progettazione come l'assenza alla completezza di Turing. Nel 2013 è stato introdotto, con un whitepaper [7] prodotto da Vitalik Buterin, Ethereum, una blockchain generica Turing-completa, con l'obiettivo di consentire la creazione di smart contract e applicazioni decentralizzate (DApps). Uno smart contract è un programma che viene eseguito dall'EVM (Ethereum Virtual Machine) sulla propria blockchain. Essendo una tecnologia Turing-completa distribuita, gli sviluppatori hanno dovuto trovare una soluzione per affrontare un possibile flood del sistema derivante dal problema dell'arresto o da un attacco intenzionale. Questi sono stati parzialmente aggirati con l'introduzione del Gas, un'unità di misura per il costo computazionale necessario per eseguire le operazioni. La natura distribuita e l'impossibilità teorica di incorrere al problema dell'arresto rende Ethereum una sfida dal punto di vista della sicurezza. Come evidenziato dal report "The 2023 Crypto Crime Report" [8] prodotto da Chainalysis, il 2022 è stato l'anno più importante di sempre per l'hacking di criptovalute, con 3,8 miliardi di dollari rubati, principalmente dai protocolli DeFi. L'impiego di strumenti automatizzati per la ricerca di vulnerabilità risulta cruciale per estendere il raggio di azione e controllo in quanto spesso errori e bug si traducono in significative perdite di denaro.

6 Creazione property-based tester

Un fuzzer è uno strumento che genera input casuali o semi-casuali per testare la robustezza di un'applicazione o di un sistema informatico. Viene utilizzato per scoprire bug o errori di programmazione eseguendo una serie di test. Un property-based tester è un fuzzer che effettua una verifica automatica sulla base di proprietà o invarianti, definite in

forma di dichiarazioni matematiche o logiche, che per il programma in analisi dovrebbero essere sempre soddisfatte. Per entrare nel dettaglio facciamo un esempio del funzionamento: consideriamo un programma che permette ad un utente A di inviare un importo x di token a un altro utente B. Innanzitutto, lo sviluppatore scrive alcune funzioni chiamate invarianti che dovrebbero sempre restituire true se il programma è stato implementato correttamente. L'invariante definisce che l'invio di token avvenga solo quando il saldo iniziale di A risulta superiore o uguale all'importo richiesto x. A questo punto il fuzzer, prende come input sia il software da controllare che l'invariante, generando migliaia di test casuali e verificando ad ogni test che l'invariante sia rispettata.

Nel contesto degli smart contract, i test generati sono in realtà sequenze di transazioni inviate al contratto ed eseguite su di una testnet. Dopo ogni singola transazione, tutti gli invarianti vengono verificati. Se al termine della verifica tutte le proprietà rispondono positivamente, lo stato della blockchain locale viene ripristinato e un'altra sequenza di transazioni verrà inviata al nodo locale. Questo ciclo verrà ripetuto molte volte finché non verrà rilevato un riscontro negativo negli invarianti, o fino alla generazione di un numero di test sufficiente a dimostrare la qualità del codice. Se si scopre che alcuni invarianti sono fragili, si ricorre alla fase finale chiamata "shrinking" o restringimento. Essa, è facoltativa ma molto utile per comprendere l'origine del problema. L'obiettivo del restringimento è quello di trovare il caso di test più semplice che causa la rottura dell'invariante, ad esempio: se un intero senza segno è causa della rottura, il restringimento troverà l'intero più piccolo responsabile dell'attivazione. altro esempio potrebbe essere quello di trovare la sequenza più breve di transazioni responsabili della falsificazione dell'invariante, in un caso in cui siano necessarie diverse chiamate.

Il progetto è stato scritto interamente in Python, un linguaggio ad alto livello, noto per la sua versatilità. Per la generazione dei test ho adottato Hypothesis che permette la descrizione di come dovrebbero apparire gli input invece di specificarli direttamente. Le transazioni sono state convalidate su una testnet eseguita in locale; Anvil dal framework di Foundry. Questa scelta si è rivelata molto efficace, superando l'opzione più adottata, Ganache.

In questo lavoro ho proposto il mio property-based tester scritto in Python un framework fuzzing testato per rilevare 5 tipi di vulnerabilità sugli smart contract. Il nostro esperimento effettuato con contratti estratti da challenge e da esempi del mondo reale tra cui, DAO e il Parity Wallet bug, ha mostrato che la strategia di generazione di input tramite la definizione di invarianti sia efficace per rilevare vulnerabilità con una buona precisione. Il confronto con Echidna, che implementa la medesima tecnica, ha mostrato un tasso di precisione simile nella ricerca di vulnerabilità nelle classi esaminate ma, con una significativa lentezza nel confronto ed una peggiore user-experience. Tutto questo può essere compensato dalla versatilità e l'implementabilità di Python.

7 Bibliografia

[1] A .Groce, J. Feist, G. Grieco, M. Colburn. "What are the actual flaws in important smart contracts (and how can we find them)?". In: Financial Cryptography and Data Security: 24th International Conference (2020)

[2] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: Decentralized business review (2008)

[3] W. S. Stornetta e S. Habere. "How to time-stamp a digital document" (1991)

[4] L. Lamport. "The weak Byzantine generals problem" In: Journal of the ACM (JACM) (1983)

[5] R. Merkle. "A certified digital signature" In: Conference on the Theory and Application of Cryptology (1989)

[6] C. Dwork e M. Naor "Pricing via processing or combatting junk mail" In: Annual international cryptology conference (1992)

[7] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". In: white paper 3.37 (2014)

[8] Chainalysis. "The 2023 Crypto Crime Report" URL: https://go.chainalysis.com/2023-crypto-crime-report.html (2023)