IMT SCHOOL FOR ADVANCED STUDIES LUCCA

VeriOSS smart contract development

Borsa di ricerca finanziata con i fondi del progetto PAI 2018 *VeriOSS: a security-by-smart contract verification framework for Open Source Software* - P0137

VERIOSS: CHALLENGE-RESPONSE INTERACTION V.2

FRANCESCO MUCCI
francesco.mucci@imtlucca.it, francesco.mucci@stud.unifi.it

ABSTRACT

Andiamo a descrivere in maniera sintetica le modifiche fondamentali che saranno apportate nella nuova versione della *challenge-response interaction* (cuore del processo di *Bounty Claim* della piattaforma *VeriOSS*). In questa nuova versione andremo a spostare off-chain parte dei dati e delle computazioni affidate al `ChallengeReward` contract (lo smart contract responsabile del processo di partial rewarding) in modo da ridurne il costo in termini di gas points. Per fare ciò, ci appoggeremo ad un data storage distribuito (IPFS) e ad un oracle service (Provable).

CHALLENGE-RESPONSE INTERACTION V.2

Rispetto alla versione descritta nel paper *"VeriOSS: using the Blockchain to Foster Bounty Programs"*, spostiamo off-chain:

- tutte le `debug-info` fornite dal BH;
- le computazione relative alle funzioni `solve` e `decommit`.

Le `debug-info` verranno caricate su *InterPlanetary File System (IPFS)*[1] node e saranno recuperabili da BI e dall'oracolo attraverso il loro IPFS-hash (multihash codificato con Base58); tale multihash potrà essere salvato come state variable in `Bounty` e `ChallengeReward` smart contracts.

1 Nel prototipo si potrebbe anche non usare IPFS, ma un altro data-storage-service (anche centralizzato).

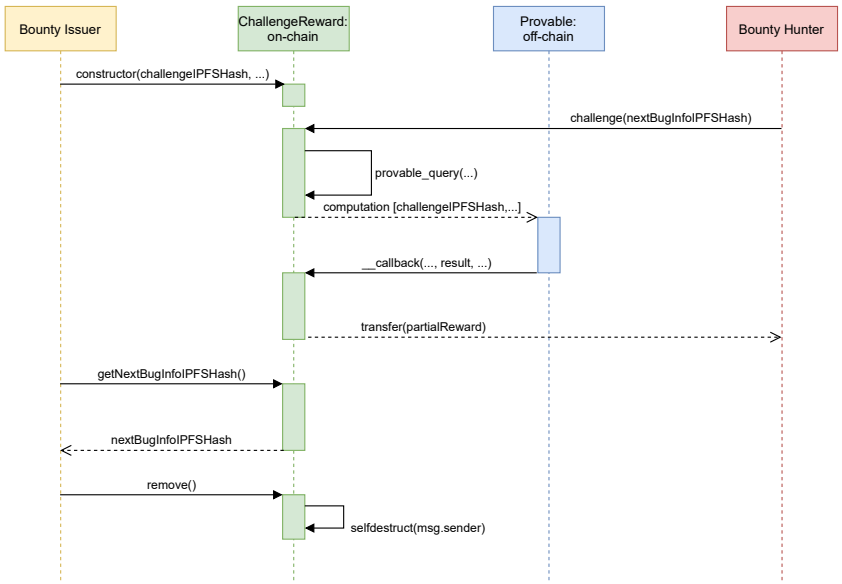

Figura 1: Sequence diagram della challange-response interaction

Le computazioni relative alle funzioni `solve` e `decommit` saranno delegate a *Provable oracle service*. Per far ciò, BI dovrà caricare su IPFS un archivio zip contenente il binario o lo script da eseguire, le eventuali dependecies e il Dockerfile che descrive l'execution context; il multihash di tale archivio sarà fornito dal BI al momento del deploy del `ChallengeReward` contract e verrà salvato come sua state variable.

BH invocherà la funzione `challenge` e le passerà l'encrypted-IPFS-hash della `next-debug-info`; tale hash sarà stato cifrato usando la chiave pubblica di Provable.

La funzione `challenge` interrogherà Provable attraverso una *nested-query* (una query che si appoggia ad una sub-query): la sub-query chiederà a Provable di decifrare l'encrypted-IPFS-hash usando la propria chiave privata e di usarlo per recuperare la `next-debug-info` dalla IPFS network; la query principale sarà di tipo `"Computation"` ed avrà come argomenti l'IPFS-hash dell'archivio e la `next-debug-info` recuperata tramite la subquery. Facendo ciò, la funzione `challenge` delegherà il check della soluzione della challenge ed il check dell'`obfuscated-bug-trace` a Provable.

A patto che il contratto abbia fondi a sufficienza per coprire le varie fees,

Provable eseguirà, in docker container su AWS virtual machine, il binario/script contenuto nell'archivio (gli argomenti necessari saranno passati come variabili d'ambiente); il risultato della computazione sarà restituito, sotto forma di stringa, attraverso la transazione che invoca la funzione `__callback`[2]. La stringa di risultato sarà:

- in caso di fallimento, `"NOT PASSED"`;
- in caso di successo, l'encrypted-IPFS-hash della `next-debug-info` (questa volta cifrato con la chiave pubblica di BI).

La funzione `challenge` salverà in una variabile di stato il risultato e, unicamente in caso di successo, trasferirà il `partial-reward` all'`address` dell'BH. Il BI potrà recuperare il risultato leggendo la state variable pubblica.

L'aver cifrato l'IPFS-hash della `next-debug-info` con la chiave pubblica di Provable ci fornirà una doppia protezione:

- la confidenzialità dell'IPFS-hash della `next-debug-info` sarà garantita nonostante sia possibile eseguire uno scrutinio della blockchain ethereum (che è pubblica);
- saremo protetti da replay-attack attraverso un meccanismo interno a Provable: il primo contratto che lo interroga con una certa encrypted-query ne diventa il proprietario; ogni altro contratto che eseguirà successive interrogazioni con la stessa stringa riceverà un empty result.

In caso di superamento della challenge, la stringa di risultato verrà cifrata con la chiave pubblica di BI; in tal modo, avremo la garanzia che la confidenzialità dell'informazione sia garantita nonostante l'uso di blockchain pubblica; tuttavia, è importante ricordare che, in ogni caso, la `next-debug-info` sarà immagazzinata in chiaro in IPFS node.

Per risolvere efficacemente il problema della *early-disclosure* del bug, si potrebbe andare a caricare su IPFS due versioni della `next-debug-info`: la prima cifrata con la chiave pubblica di Provable e la seconda con quella del BI; tuttavia, è necessario ulteriore studio al fine di vagliare la fattibilità della soluzione.

2 Sarà necessario che tale funzione venga definita nel `ChallengeReward` contract.