SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Iva Kustura

FALLBACK FUNKCIJE U SOLIDITY JEZIKU

SEMINARSKI RAD

Varaždin, 2024

Sadržaj

1. Uvod

Ovim projektom ću objasniti što je Blockchain, navesti osnovne karakteristike te prikazati njegovu primjenu u različitim industrijama. Postoje različite vrste blockchaina (privatni, javni i hibridni), a neke od njih će biti definirane u nastavku. Nakon toga slijedi pregled Remix Ethereum okruženja u kojemu se razvijaju pametni ugovori. Programski kod koji koristimo za definiranje ugovora je Solidity čije će osnovne karakteristike također biti prikazane. Uz osnovne karakteristike, prikazat ću usporedbu s ostalim programskim jezicima kako bi lakše razumjeli osnovne koncepte ovog programskog jezika.

Praktični dio će pokriti primjenu fallback funkcija koje se koriste za primanje Ethera te u slučaju kada niti jedna druga funkcija ne odgovara pozivu. Postoji niz karakteristika ovih funkcija te će biti navedene i objašnjene u nastavku ovog poglavlja.

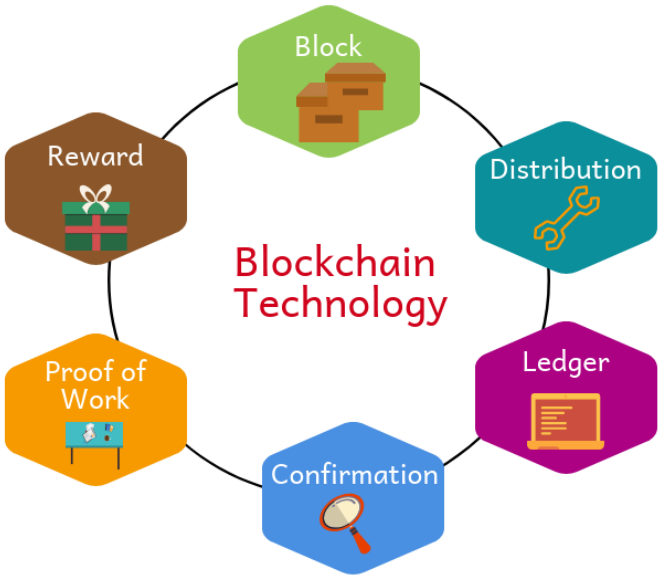
Nakon teoretskog objašnjenja, prikazat ću primjer primjene te objasniti kako programski kod funkcionira. Vlastiti primjer implementacije fallback funkcija bit će naveden nakon ovog dijela. Na primjeru ću prikazati kako poslati Ethere s jednog ugovora na drugi te zabilježiti osnovne podatke o transakciji.

Postoji niz rizika koje nose fallback funkcije jer može doći do gubitka Ethera u slučaju da implementacija sadrži sigurnosne propuste. Prikazat ću jedan primjer gdje napadač može iskoristiti takve ranjivosti te nakon toga navesti i objasniti koje su metode sprječavanja zlouporabe te načini smanjenja rizika. Ukratko će biti objašnjen alat SolidityScan koji nudi statičku analizu koda i na taj način pomaže u smanjenju sigurnosnih propusta i drugih grešaka.

2. Blockchain

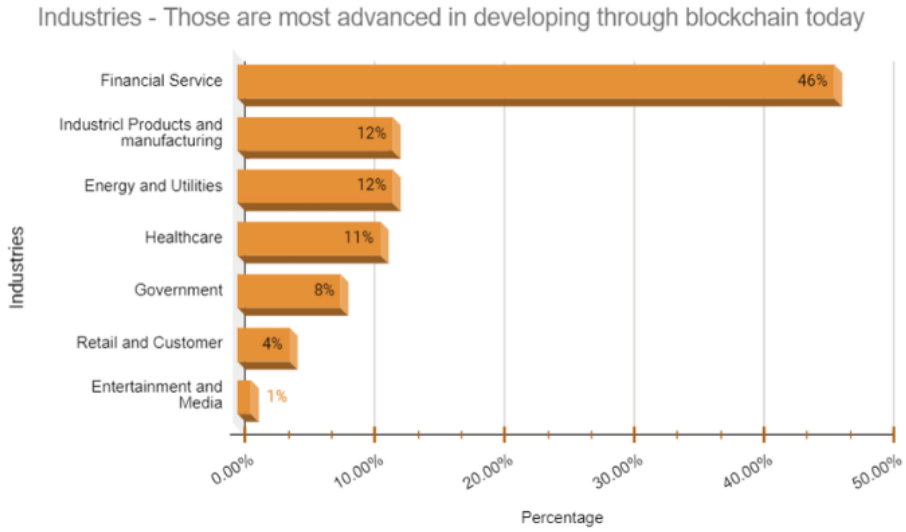
Blockchain možemo smatrati registom koji služi za praćenje transakcija i imovine u svijetu poslovanja. Prednost blockchaina je što umanjuje troškove trgovanja za sve vrste imovine (materijalna i nematerijalna) [1] . Trenutno postoji više od 23 000 sustava kriptovaluta te je dokazano da je način pohrane podataka o transakcijama na blockchainu pouzdan. Dijeljenje informacija putem blockchaina je izvrsno jer se može doći do informacija koje su točne i neizbrisive, dok s druge strane, pristup informacijama imaju samo ovlašteni članovi mreže. Transakcije se obavljaju putem pametnih ugovora koji sadrže pravila, a pohranjuju se na blockchain-u.

Slika ispod prikazuje osnovne karakteristike Blockchain tehnologije. Reward se odnosi na sustav nagrađivanja rudara kada uspješno dodaju novi blok u lanac. Takva nagrada rudarima daje određeni iznos kriptovalute,a naziva se „block reward“. Na taj način se održava motivacija rudara kako bi i dalje nastavili pružati svoju računalnu snagu. Danas to više nije posao koji može obavljati osobno računalo jer zahtjeva puno veću računalnu snagu i nije isplativo. „Proof of Work“ se odnosi na provjeru rada rudara u kojem ostali članovi mreže mogu provjeriti ispravnost rada. Potvrda (eng Confirmation) se odnosi na transakcije koje čekaju na verifikaciju od strane rudara. Vrijeme čekanja na potvrdu ovisi o težini rudarenja, naknadi transakcije i slično. Evidencija transakcija sadrži sve transakcije i događaje koji su zabilježeni te imaju svojstvo nepromijenivosti. Osim toga, transakcije su javne i vidljive svim sudionicima mreže. [2]



Slika 1 Blokchain tehnologija

Grafikon ispod prikazuje u kolikom se postotku koristi Blockchain tehnologija u različitim industrijama. Najveća uporaba je zabilježena u financijskim uslugama gdje se 46% temelji na blockchainu. Postoji primjena i u zdravstvu jer je u SAD-u uveden zakon da bolnice moraju implementirati elektroničke medicinske zapise (EMR). Što se tiče nekretnina, ova tehnologija bi smanjila količinu papirologije jer bi se svi dokumenti mogli pohraniti uz manje troškova i posla.



Slika 2 Indrustrije koje koriste Blockchain

Postoje različite vrste blockchain-a obzirom na privatnost. Javni blockchain je prva vrsta koju povezujemo s Bitcoin-om. Ovakva vrsta je otvorena za sve, a sve transakcije su zabilježene. Ne postoje zabrane jer jer svi koji se pridruže imaju pravo čitanja i pisanja bez potrebe za autorizacijom. Javni primjeri blokchaina su Bitcoin, Ethereum, NEO …

Privatni blockchain su centralizirani, za razliku od javnih. Dakle, centralna vlast određuje tko može čitati, pisati te sudjelovati u aktivnostima mreže. Ovakve mreže su zbog navedenog poznate kao „permissioned blockchain networks“. Imaju svojih prednosti koje su veća prilagodljivost te korištenje za pohranu osjetljivih podataka. Upravo iz tog razloga, članovi se moraju autentificirati te ne mogu biti anonimni. Primjeri ovakvih lanaca su MultiChain, Hyperledger, i Corda.

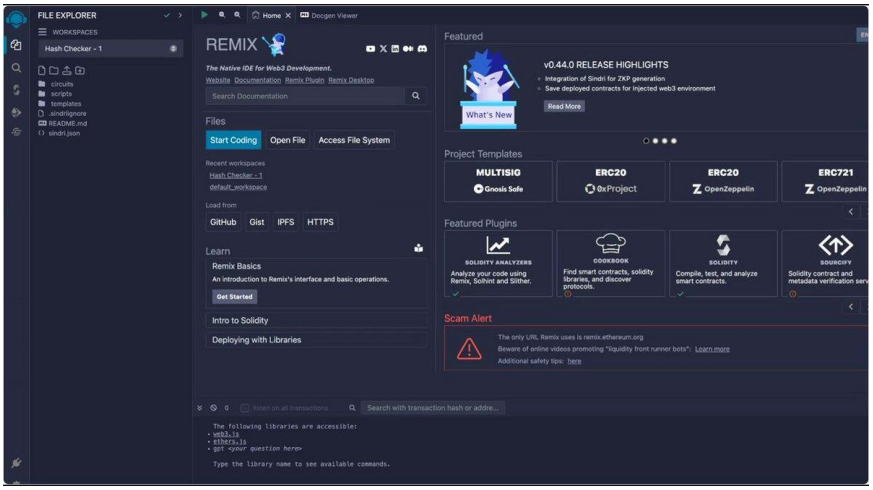
Hibridni lanci nude elemente javnih i privatnih tako da nude dobre karakteristike dviju vrsta. Transakcije nisu javne, ali se mogu provjeriti u slučaju da je to potrebno. Korisnici otkrivaju svoj identitet drugim stranama kada sudjeluju u transakcijama. Primjeri ovakvih lanaca su Dragonchain, XinFin i Ripple. [4]

3. Remix Ethereum

Remix Ethereum IDE je integrirano razvojno okruženje za razvoj pametnih ugovora na Ethereum blockchainu. Može se koristiti iz preglednika jer je „web-based“ alat koji ne zahtjeva posebnu instalaciju na računalu. Svoje programske datoteke moguće je organizirati u mape kako bi se mogli lakše snalaziti. Sve datoteke je moguće pregledavati i ažurirati te se neometano prebacivati iz jedne u drugu. Postoji opcija i za „debug“ kako bi mogli bolje razumjeti što se događa u kodu i predvidjeti moguće pogreške.

Remix Ethereum podržava različite verzije jezika Solidity što omogućuje razvoj ugovora koji su kompatibilni s različitim verzijama Ethereum blockchainom. Preko ugrađenog web3 preglednika omogućena je transakcija s Ethereum blockchainom. Pisanje koda olakšano je kroz Ethereum Editor koda koji naglašava bojama određene dijelove te nudi automatsko dovršavanje započetog koda. Na slici ispod se nalazi Remix sučelje gdje možemo vidjeti datoteke s lijeve strane te gumbe koji vode na komiliranje, „deploy“ koda, debugger, pretraživanje datoteka. U slučaju da se datoteka ne može kompilirati, ponuđena je pogreška i mogućnost kopiranja iste te upit za ChatGPT. Ispod datoteke se nalazi konzola u koju možemo zapisivati „logove“ kako bismo provjeravali ispravnost koda. Osim toga, moguće je kreirati jedinične testove kako bi se provjerilo ponašanje koda. [5]

Prilikom kompiliranja koda možemo odabrati verziju kompilera, ali s oprezom da verzija ne bude manja ili veća od raspona kojeg smo naveli u programskom kodu. Za „deploy“ i pokretanje transakcija možemo odabrati okruženje, račun. U slučaju da testiramo fallback i receive funkcije možemo postaviti vrijednost Ethera za slanje.



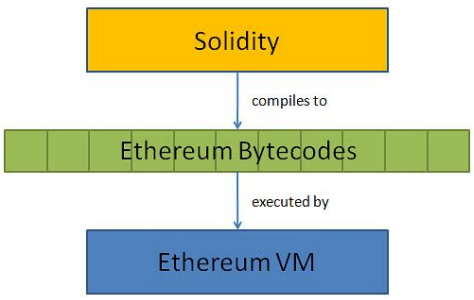
Slika 3 Ethereum sučelje

4. Pregled Solidity jezika

Solidity programski jezik je objektno orijentiran i koristi se za kreiranje pametnih ugovora na Blockchain platformama. Njegova glavna svrha je kreirati ugovore koji automatiziraju i izvršavaju digitalne sporazume na blokchain-u. Omogućene su poslovne aktivnosti koje uključuju plaćanja, primanje novca, osiguranja i slično [6].

Postoji jako puno sličnosti s drugim programskim jezicima kako što su C i C++ te je iz tog razloga jednostavan za razumjeti i naučiti, pogotovo ako postoji iskustvo u navedenim jezicima. Osim toga, postoje sličnosti i sa drugim programskim jezicima kao što su JavaScript i Python. Što se tiče JavaScripta, ranije je postojala veća sličnost, ali je trenutno značajna sličnost korištenje iste riječi za kreiranje funkcija „*function*“. Primjer sličnosti s programskim jezikom Python je korištenje riječi „super“ koja se koristi kako bi se mogli referencirati na roditeljsku klasu. U Solidity-ju se nasljeđivanje koristi u kontekstu ugovora u slučaju kada želimo pristupiti funkcijama i podacima iz roditeljskog ugovora. Trenutno je osnovni jezik na Ethereum platformi koju koristi za kompiliranje koda te je široko rasprostranjen jezik u „blockchain“ razvojnoj industriji [7].

Važno je razumjeti kako Solidity funkcionira kako bi programiranje postalo razumljivije i brže. Pametni ugovori su programi koji se izvršavaju na Ethereum virtualnom stroju (eng. Ethereum Virtual Machine - EMV). Pametne ugovore možemo povezati s klasama u drugim programskim jezicima – mogu imati funkcije, unutarnja stanja i javne deklaracije. Kreirani ugovori se prevode u bytecode kako bi ih virtualni stroj mogao pročitati. Nakon što se ugovori kompiliraju, izgledaju jednako kao ugovori napisani na nekom drugom jeziku kao što je Vyper. Važno je naglasiti da se ugovor ne može ažurirati nakon što postane dio blokchain-a. Iz tog razloga, potrebno je dobro razmotriti koja verzija Solidity-ja se koristi prilikom programiranja. Problemi se javljaju kada se koristi starija verzija ili nepodržana jer može doći do nekompatibilnosti ili sigurnosnih rizika. Slika 4 prikazuje ranije opisan način funkcioniranja programskog koda u Solidity-ju s Ethereum virtualnim strojem.



Slika 4 Solidity i Ethereum VM

5. Fallback funkcije

Fallback funkcije su posebne funkcije u Solidity jeziku koje se pozivaju u trenutku kada ne postoji funkcija u pametnom ugovoru koja bi odgovarala pozvanoj fukncji ili ako poziv funkcije ne sadrži nikakav podatak. Moguće je dodijeliti samo jednu funkciju bez naziva koja će se izvršavati svaki puta kada ugovor primi Ether bez podataka [8]. U ovom poglavlju ću prikazati kako se koriste fallback funkcije te prikazati svoj primjer implementacije.

5.1. Svojsta fallback funkcija

Fallback funkcije moraju biti označene kao „payable“, ako ne postoji takva funkcija ugovor ne može primiti Ether i dogodit će se pogreška. Ranije je spomenuto da ih se poziva u dva uvjeta što znači da se ne pozivaju eksplicitno već u određenim slučajevima. Osim toga, fallback funkcije ne mogu vraćati vrijednost jer su dizajnirane kako bi izvršile određene operacije. Ova karakteristika je važna jer se pozivaju u specifičnim slučajevima pa ne bi imalo smisla da vraćaju neke podatke. Iako ne primaju nikakve podatke, njihova logika može biti vrlo složena jer mogu obrađivati transakcije, ažurirati stanje ugovora i slično. Postoji ograničenje od 2300 goriva (eng. gas) kada je pozvan „transfer“ ili „send“. Ovo ograničenje postoji kako bi se spriječilo prekoračenje goriva prilikom izvršavanja funkcija unutar ugovora. Ne mogu se proslijeđivati podatci prilikom poziva jer fallback funkcije ne mogu imati argumente. Nužno je označiti ih kao „external“ kako bi bile vidljive izvan ugovora ili iz blockchina [9].

Nakon 0.6 verzije podijeljene su u dva dijela : „receive“ i „fallback“. „Receive“ funkcija se koristi prilikom prilikom čistog prijenosa Ethera koji se šalje na pametni ugovor. Ovo je već objašnjen slučaj kada se pošalje samo Ether, funkcija se aktivira te nakon toga obrađuje poslani Ether. Fallback funkcija se koristi za rukovanje Etherom i u slučaju kada poziv ne odgovara niti jednoj drugoj funkciji ugovora. Predstavlja standardni način korištenja fallback funkcija pa zbog toga ovakve funkcije mogu rukovati s Etherom bez pridruženih podataka kao i receive funkcija. Slika 5 prikazuje sintaksu receive i fallback funkcija. Obje funkcije su označene s „external“ kako bi im se moglo pristupiti te imaju oznaku „payable“ kako bi ugovor mogao primiti Ether.

```
fallback() external payable {
    // Something
}
receive() external payable {
    // Something
}
```

Slika 5 Receive i fallback funkcije

5.2. Primjeri upotrebe

Prvi programski primjer sastoji se od dva pametna ugovora : „Bank“ i „You“. Ugovor *Bank* sadrži funkciju *fallback()* koja se aktivira kada na ugovor Bank dođe Ether koji nema podataka ili poziva funkcije. Tijelo funkcije sadrži događaj koji sadrži adresu pošiljatelja, vrijednost Ethera te poruku „JustFallbackwithFUnds is called“. Druga funkcija ugovora *Bank* je *receive()* koja emitira događaj *ReceiveFallback* s adresom pošiljatelja i vrijednošću Ethera.

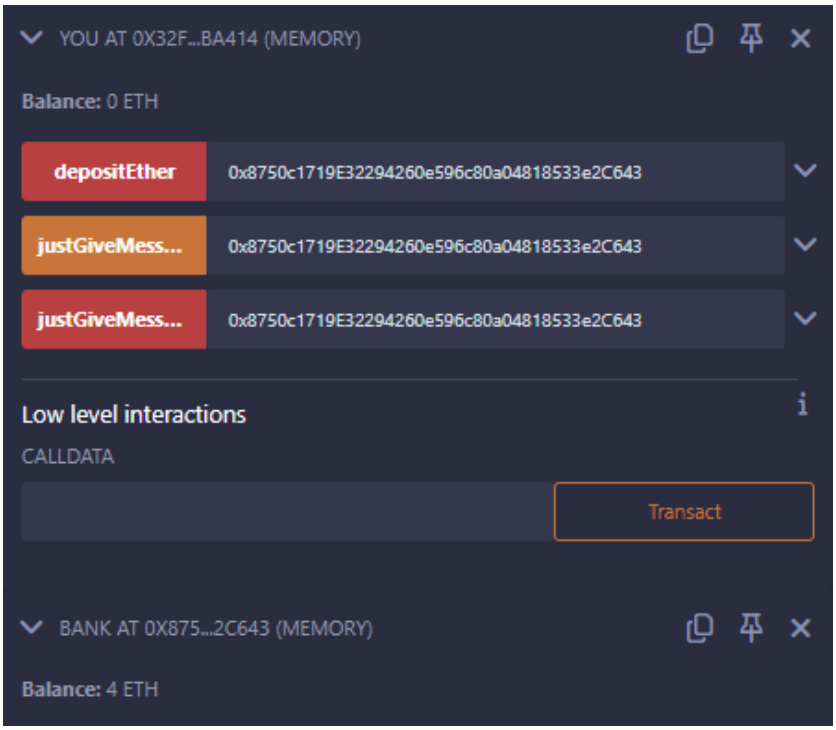
Ugovor „You“ kroz funkciju *depositEther()* prima adresu pametnog ugovora *Bank* u argumentu te šalje Ethere na adresu. Nakon kompiliranja oba ugovora, potrebno je kopirati adresu Bank ugovora kako bi se funkcijama drugog ugovora mogao dati argument adrese. Ostale funkcije ugovora (*justGiveMessage* i *justGiveMessageWithFunds*) pozivaju funkcije s nazivom „Hi“. Ako su pozivi uspješni, emitira se događaj, inače se vrati iznimka.

```
pragma solidity >=0.7.0 <0.9.0;

contract Bank {
    event JustFallback(address _from, string message);
    event ReceiveFallback(address _from, uint256 _value);
    event JustFallbackWithFunds(address _from, uint256 _value, string
message);

    fallback() external payable {
        emit JustFallbackWithFunds(msg.sender, msg.value, "Fallback funkcion
is called");
    }
    receive() external payable {
        emit ReceiveFallback(msg.sender, msg.value);
    }
}

contract You {

    function depositEther(address payable _to) public payable {
        _to.transfer(msg.value);
    }

    function justGiveMessage(address _to) public {
        (bool success, ) = _to.call("HI");
        require(success, "Failed");
    }

    function justGiveMessageWithFunds(address payable _to) public payable {
        (bool success, ) = _to.call{value:msg.value}("HI");
        require(success, "Failed");
    }
}
```

Programski primjer je preuzet s [10]

Nakon „deploy-a“ potrebno je unijeti adresu banke kako bismo mogli testirati funkcije. Funkcija *depositEther()* je poslala zadanu vrijednost Ethera ugovoru *Bank*. Ako bismo postavili

vrijednost Ethera prilikom poziva funkcije koja nije označena kao payable, dobili bismo pogrešku. Takav primjer je funkcija *justGiveMessage().* Sljedeća funkcija šalje poruku, ali i vrijednost u Etherima. Pozivom funkcije *depositEther(),* poziva se receive funkcija ugovora *Bank,* dok ostale funkcije pozivaju fallback funkciju. U zapisima se ispisuje poruka u slučaju da je pozvana fallback funkcija.



Slika 6 Ugovori Bank i You

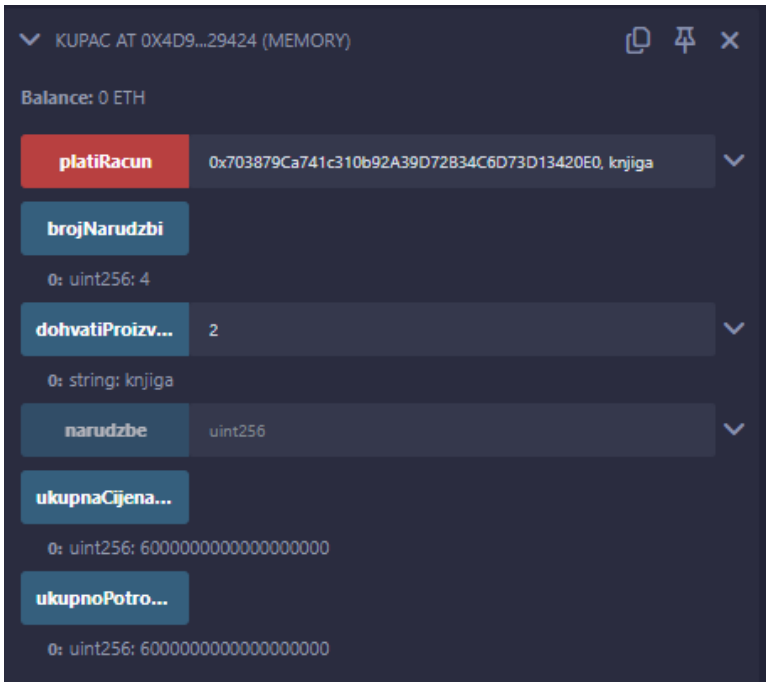
5.3. Implementacija fallback funkcija

Ovaj primjer prikazuje sustav za slanje Ethera između dva pametna ugovora : *Prodavac* i *Kupac. Prodavac* prima Ethere koji se šalju kroz funkciju *receive()* koja emitira događaj *primljenoPlacanje.* Kako bismo provjerili stanje Ethera na ugovoru koristimo *dohvatiStanje(). Kupac* ugovor predstavlja klijenta koji šalje Ethere metodom *platiRacun*. Funkcija *platiRacun ()* sprema objekt tipa *racun* u listu narudžbi te se dodaje vrijednost ukupnoj cijeni narudžbe. Adresa prodavača, odnosno adresa ugovora i proizvod su definirani u parametrima funkcije. Nakon toga, Etheri se prebacuju na adresu prodavača.

```
pragma solidity >=0.6.0 < 0.9.0;
contract Prodavac{
    event primljenoPlacanje(address sender, uint256 amount);
    receive() external payable{
        emit primljenoPlacanje(msg.sender, msg.value);
    }
```

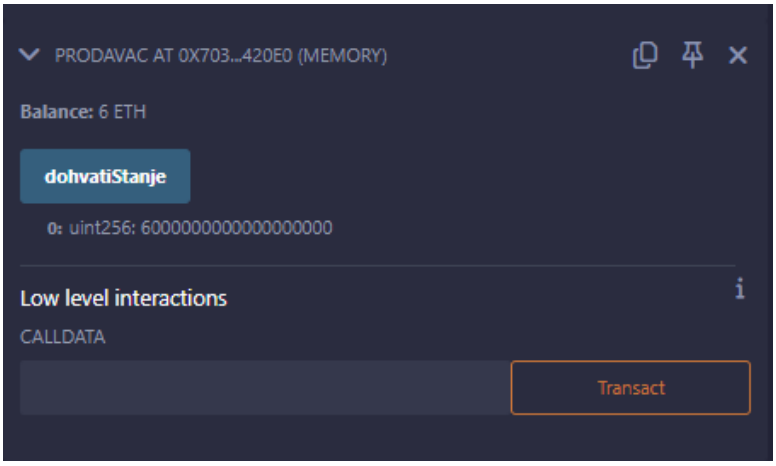
```
    function dohvatiStanje() public view returns(uint){
        return address(this).balance;
    }
}
contract Kupac{
    uint public ukupnaCijenaNarudzbi;
    struct racun{
       address adresaKupca;
       string proizvod;
       uint cijena;
    }
    racun[] public narudzbe;
    function ukupnoPotroseno() public view returns(uint) {
        return ukupnaCijenaNarudzbi;
    }
    function brojNarudzbi() public view returns(uint) {
        return narudzbe.length;
    }
    function platiRacun(address payable prodavac, string memory proizvod)
public payable{
        narudzbe.push(racun(msg.sender, proizvod, msg.value));
        ukupnaCijenaNarudzbi += msg.value;
        prodavac.transfer(msg.value);
    }
     function dohvatiProizvod(uint index) public view returns ( string
memory){
        racun memory dohvaceniRacun = narudzbe[index];
        return  (dohvaceniRacun.proizvod);
    }
}
```

Slika ispod prikazuje ugovor *Kupac* koji može slati Ethere na način da u funkciju *PlatiRacun()* unese parametre – adresa prodavača i naziv proizvoda. Kako bi slanje Ethera bilo više od nula, potrebno je postaviti „VALUE“ na broj Ethera koji želimo poslati. U ovom primjeru, možemo vidjeti da su kreirane 4 naružbe, da je potrošeno ukupno 6 Ethera te da je račun pod rednim brojem 2 knjiga.



Slika 7 Ugovor Kupac

Ispod se nalazi ugovor prodavača koji nakon svih provedenih transakcija ima stanje 6 Ethera što možemo vidjeti iz funkcije *dohvatiStanje(),* ali i iz vrijednosti *Balance.*



Slika 8 Ugovor Prodavac

Slika ispod prikazuje zapise o transakciji nakon slanja i pozivanja *receive()* funkcije. Možemo vidjeti da je transakcija uspješno provedena. U zapisima se također nalazi adresa pošiljatelja te vrijednost poslanih Ethera. Adresa prodavača i naziv proizvoda nalaze se u „decoded input“ dijelu.

Slika 9 Zapisi transakcije

6. Sigurnosni rizici

Postoje sigurnosni rizici u fallback funkcijama koje napadači mogu iskoristiti. U navedenom primjeru, napadač može najprije pozvati funkciju *addAllMoney()* s vlastitom adresom kako bi se registrirao u *bilances.* Nakon toga, može kreirati ugovor s fallback funkcijom i pozvati funkciju *withdrawAllMoney()* s određenim iznosom. Pozvat će se fallback funkcija linijom *msg.sender.call.value(Totalamount)(")* koja će ponovno pozvati funkciju *withdrawAllMoney()* i na taj način će se spriječiti smanjenje stanja msg.sender-a. Napadač izaziva rekurzivne pozive koji mu omogućuju zadržavanje vlastitih sredstava.

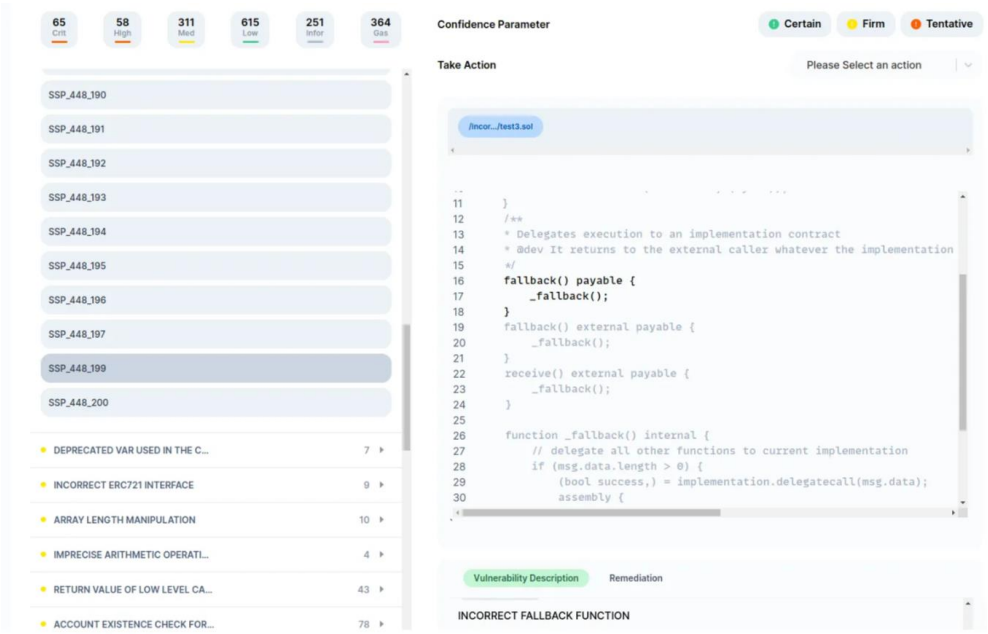
```
contract fallbackVulnerability {

    mapping (address => uint) private balances;

    function addAllMoney (address Useraccount) public payable {
        balances [Useraccount] = msg.value;
    }

    function withdrawAllMoney (uint Totalamount) public {
        if (balances [msg.sender]>= Totalamount) {
            msg.sender.call.value(Totalamount)("");
            balances [msg.sender] -= Totalamount;
        }
    }
}
```

Preuzet programski primjer [11]

6.1. Tehnike smanjenja ranjivosti i rizika

Postoje načini kako možemo spriječiti zlouporabu i smanjiti ranjivost u fallback i receive funkcijama. Prva tehnika je postavljanje ugovora na verziju 0.6 ili više. Osim toga, receive funkcija može služiti samo za prihvaćanje Ethera i na taj način spriječiti moguće gubitke Ethera.

Druga tehnika je smanjiti fallback metode tako da one samo bilježe događaj jer one mogu koristiti samo 2300 goriva. Na taj način se sprječavaju nepredviđeni troškovi i problemi sa izvođenjem. Ako je fallback funkcija namijenjena za zapisivanje primljenog Ethera trebali bi se pobrinuti da su podatci prazni jer se fallback funkcije koriste i u slučajevima kada pozivu ne odgovara niti jedna druga funkcija. Ugovori koji pozivaju metode neće biti svjesni propusta u našim metodama pa treba oprezno i pažljivo pisati programski kod. [11]



Slika 10 Solidity Scan

Slika iznad prikazuje sučelje alata SolidityScan koji utvrđuje netočne implementacije fallback funkcija. SolidityScan je alat za statičku analizu pametnih ugovora napisanih u Solidity jeziku. Koristi se za otkrivanje potencijalnih sigurnosnih propusta i grešaka u kodu. Može biti vrlo koristan jer se problemi mogu otkriti na vrijeme i na taj način izbjeći štete koje mogu nanijeti napadači.

Alat pruža izvještaje o pronađenim problemima, opisuje greške i načine za rješavanje problema. Iako pomaže razvojnim timovima, alat ne može otkriti sve sigurnosne propuste. Kako bi razina pouzdanosti i sigurnosti bila na višoj razini, razvojni timovi mogu koristiti više alata, pratiti i ažurirati ugovore i testirati programski kod.

7. Zaključak

Blockchain tehnologija je važna zbog svojeg velikog napretka i utjecaja na različite industrije. Iz tih razloga, važno je poznavati barem osnove tehnologije, Ethereum ili nekog drugog okruženja te Solidity jezika koji omogućava razvoj pametnih ugovora.

Blockchain bilježi transakcije te ima važnu ulogu u svijetu poslovanja. Smatra se pouzdanim jer trenutno postoji velik broj kriptovaluta i transakcija koje se bilježe svakog dana. Ovlašteni članovi mogu pristupati informacijama koje su uvijek točne i neizbrisive. Postoje razlike u vrstama blockchain-a koje se najviše tiču prava članova i anonimnosti unutar mreže. Javni lanci nude slobodan pristup korisnicima i ne traže nikakav oblik autentikacije. Hibridni lanci nemaju javne transakcije, ali se one mogu provjeriti. U privatnim lancima, centralna mreža provjerava tko može pisati i čitati transakcije, te su one iz tog razloga pogodne kod osjetljivih podataka.

Programski jezik Solidity je osnovni jezik na Ethereum platformi koju koristi za kompiliranje koda. Dijeli neke karakteristike s drugim programskim jezicima kao što su Python, Java i C++ pa je njegovu sintaksu lako naučiti. Važno je još poznavati kako funkcionira. Ugovori kreirani u Solidity jeziku se se pretvaraju u bytecode te ih nakon toga virtualni stroj može pročitati. Jednom kada postane dio blockchain-a se više ne može mijenjati te je važno promisliti o verziji koja se koristi.

Fallback funkcije važan su dio programskog jezika Solidity jer omogućuju jednu od glavnih funkcionalnosti pametnih ugovora – plaćanje i primanje novca. Njihova logika ne treba biti složena iz više razloga. Prvi od njih je što mogu biti meta napada pa zbog toga moramo biti sigurni da svaka linija u tijelu fallback funkcije ne može biti zlouporabljena. Osim toga, postoji ograničenje od 2300 gasa te ne želimo probleme u izvršavanju plaćanja zbog složene logike.

Alat SolidityScan nudi analizu koda koja može pomoći u sprječavanju grešaka i sigurnosnih propusta. To, naravno, nije dovoljno. Programeri trebaju redovito ažurirati svoj kod i verzije kako bi ostao kompatibilan. Kvalitetno testiranje također pomaže u sprječavanju navedenih problema. Moguće je kreirati jedinične testove u Remix Ethereum okruženju te na taj način efikasno i pozdano provjeriti radi li kod ono što od njega očekujemo.

8. Popis literature

[1] »What is blockchain?,« IBM, [Mrežno]. Available: https://www.ibm.com/topics/blockchain. [Pokušaj pristupa 3. 5. 2024.].

[2] »INTRODUCTION TO BLOCKCHAIN TECHNOLOGY : A NEW TECHNOLOGY,« Shekshapers, 18. 6. 2018. [Mrežno]. Available: https://www.tekshapers.com/blog/Introduction-to-Blockchain-Technology-:-A-New-Technology. [Pokušaj pristupa 10. 5. 2024.].

[3] S. Kaiser, »Towards a Blockchain Based Supply Chain Management for E-Agro Business System,« 1. 12. 2020.. [Mrežno]. Available: https://www.researchgate.net/figure/Statistics-of-Industries-those-are-most-advanced-in-developing-through_fig5_347416277. [Pokušaj pristupa 10. 5. 2024.].

[4] G. O. R. Cruz, »What Is Blockchain?,« Money, 2. 6. 2022.. [Mrežno]. Available: https://money.com/what-is-blockchain/. [Pokušaj pristupa 5. 5. 2024.].

[5] »Welcome to Remix's documentation!,« Remix Ethereum, 2024.. [Mrežno]. Available: https://remix-ide.readthedocs.io/en/latest/. [Pokušaj pristupa 5. 5. 2024.].

[6] »What is Solidity Programming: Data Types, Smart Contracts, and EVM?,« Simplelearn, 26. 5. 2023.. [Mrežno]. Available: https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-solidity-programming. [Pokušaj pristupa 5. 5. 2024.].

[7] B. Werkheiser, »What is Solidity?,« Alchemy, 22. 9. 2022.. [Mrežno]. Available: https://www.alchemy.com/overviews/solidity. [Pokušaj pristupa 5. 5. 2024.].

[8] »Solidity – Fall Back Function,« GeeksforGeeks, 6. 3. 2023. [Mrežno]. Available: https://www.geeksforgeeks.org/solidity-fall-back-function/. [Pokušaj pristupa 4 5. 2024].

[9] S. Kumar, »Fallback Functions in Solidity,« 21. 2. 2023.. [Mrežno]. Available: https://dev.to/shlok2740/fallback-functions-in-solidity-13nb. [Pokušaj pristupa 5. 5. 2024.].

[10] »Fallback with Examples,« 25. 5. 2021.. [Mrežno]. Available: https://dev.to/moyedx3/fallback-with-examples-8g3. [Pokušaj pristupa 5. 5. 2024.].

[11] »Understanding Security of Fallback & Recieve Function in Solidity,« 21. 10. 2022. [Mrežno]. Available: https://blog.solidityscan.com/understanding-security-of-fallback-recieve-function-in-solidity-9d18c8cad337. [Pokušaj pristupa 10. 5. 2024.].

Izvori slika :

[1] Blokchain tehnologija (2018.). Preuzeto 5.5.2024. s :

https://www.tekshapers.com/blog/Introduction-to-Blockchain-Technology-:-A-New-Technology.

[2] Industrije koje koriste blockchain (2020.). Preuzeto 5.5.2024. s :

https://www.researchgate.net/figure/Statistics-of-Industries-those-are-most-advanced-in-developing-through_fig5_347416277

[3] Zapisi transakcije – autorski rad

[4] Solidity i Ethereum VM (2023.). Preuzeto 5.5.2024. s :

https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-solidity-programming

[5] Receive i fallback funkcije (2021.). Preuzeto 5.5.2024. s :

https://dev.to/moyedx3/fallback-with-examples-8g3

[6] Ugovori Bank i You – autorski rad

[7] Ugovor Kupac – autorski rad

[8] Ugovor Prodavac – autorski rad

[9] Zapisi transakcije – autorski rad

[10] Solidity Scan (21.10.2022) .Preuzeto 8.5.2024. s :
https://blog.solidityscan.com/understanding-security-of-fallback-recieve-function-in-solidity-9d18c8cad337

9. Popis slika