Security Analysis of Smart Contracts on Ethereum

Bachelor's Thesis in Computer Science (B.Sc.)

Frankfurt University of Applied Sciences | 2025

Yousef Ghanem

Outline

- Motivation, Objectives and Research Question
- Methodology and Approach
- Blockchain, Ethereum, and Smart Contract
- Development and Analysis (CryptoBank)
- Improvements and Re-Analysis (CryptoBankSecure)
- Results and Discussion
- Summary and Outlook

Motivation, Objectives and Research Question

Motivation

- High relevance in practice
- Interest in security & risks
- Combine theory & practice

Objectives

- Analyze the security of Smart Contracts on Ethereum
- Identify typical vulnerabilitäten in practice

Research Question

„ Which security-relevant risks occur during the development of Smart Contracts on Ethereum and how can they be effectively detected and mitigated“

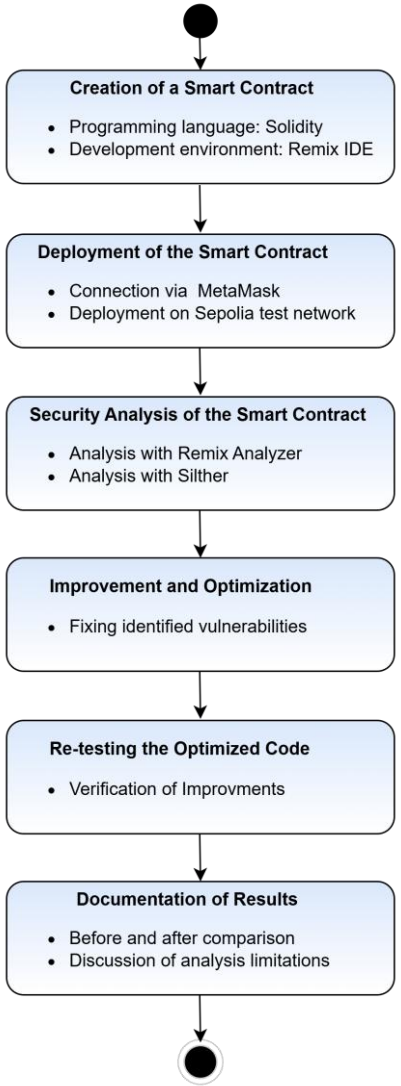
Methodology and Approach

Theory

- Blockchain, Ethereum and Smart Contracts

Practice

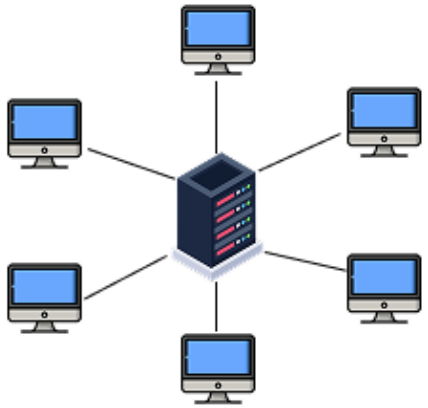
- Development with Solidity & Remix
- Deployment on Sepolia test network using MetaMask
- Security analysis with Remix and Slither
- Code improvement & re-testing



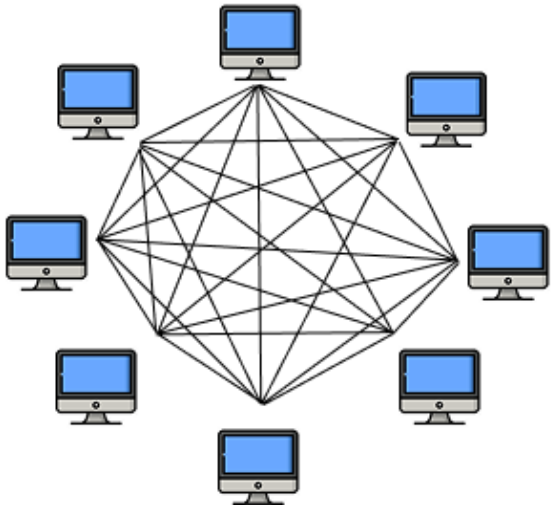
Blockchain, Ethereum and Smart Contract

Blockchain

- Decentralized data structure consisting of linked blocks
- Manipulation is hardly possible



Centralized



Decentralized

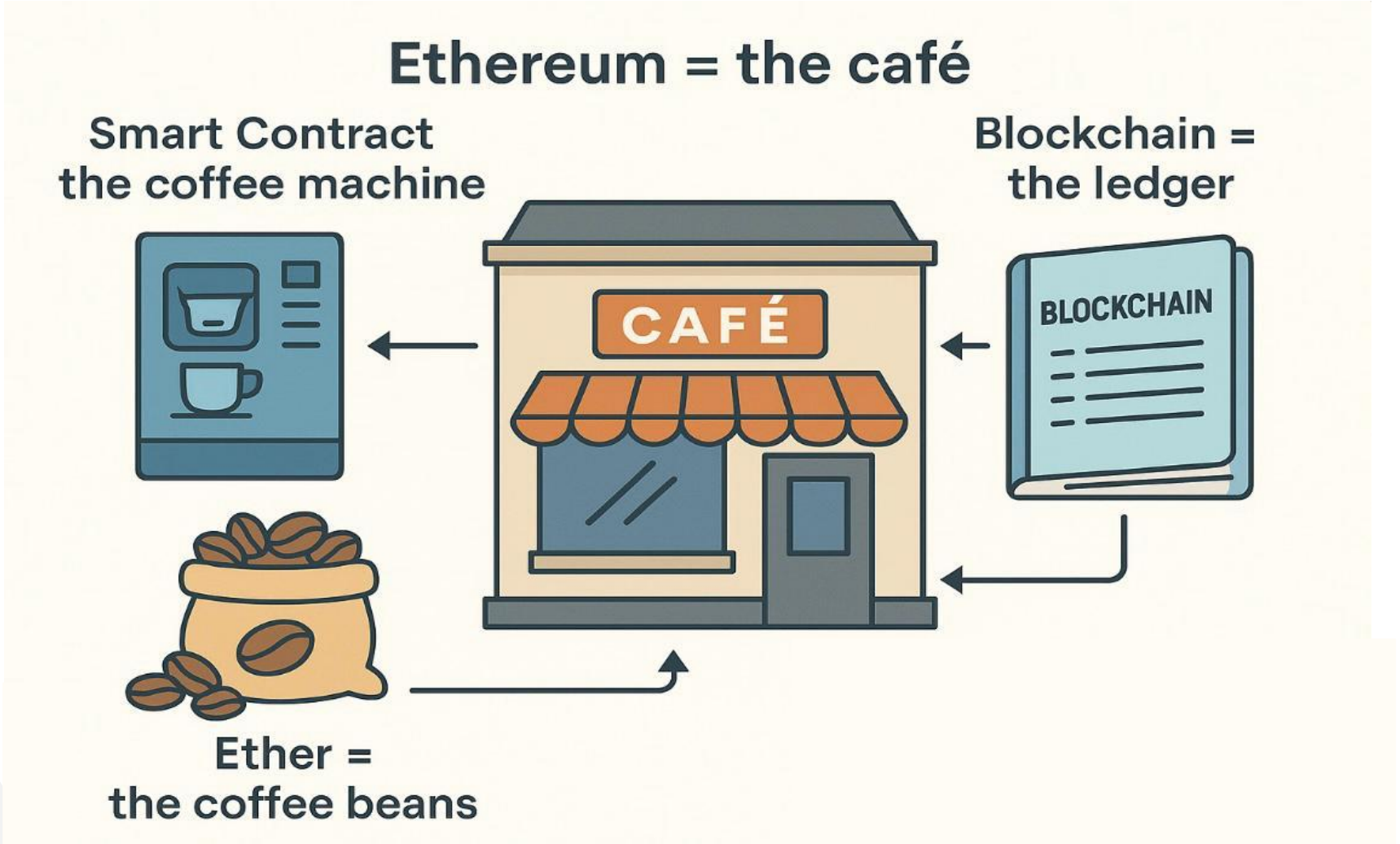
Ethereum

- Platform for decentralized applications
- Executes programs in the Ethereum Virtual Machine

Smart Contracts

- Automated contracts on the blockchain
- Immutable & deterministic

Blockchain, Ethereum and Smart Contract



Development and Analysis (CryptoBank)

Development

- Remix IDE
  - Development environment
  - Compile & deploy
- Solidity
  - Programmin language
  - Similar to JS & C++

Deployen

- MetaMask
  - Wallet & interface to the Ethereum network
  - Connect Remix with Sepolia
- Testnetz Sepolia
  - Official Ethereum test network
- Etherscan (Sepolia)
  - Blockchain explorer for Sepolia
  - Displays all transactions and contracts

Analyse

- Remix Analyzer
  - Analyse plugin
- Slither
  - External analysis tool (terminal)
  - Detailed code review for security vulnerabilities

Development and Analysis (CryptoBank)

```
contract CryptoBank {


    // hier werdem die Attribute definiert
    address public besitzer;
    mapping(address => uint) public kontostand;


    // Events für Transparenz und Analyse
    event eingezahlt(address indexed benutzer, uint betrag);
    event ausgezahlt(address indexed empfaenger, uint betrag);
    event etherEmpfangen(address absender, uint betrag);
    event FallbackGenutzt(address absender, uint value, bytes data);
```

```
    constructor() {
        besitzer = msg.sender;
    }
```

```
    // funktion für die Auszahlung: Diese Funktion wird nur von Besitzer ausgeführt
    function auszahlen(address payable _empfaenger, uint _betrag) public {
        require(msg.sender == besitzer, "Sie sind nicht der Besitzer!");
        require(kontostand[_empfaenger] >= _betrag, "Kontostand ist nicht genug!");
        kontostand[_empfaenger] -= _betrag;
        _empfaenger.transfer(_betrag);
        emit ausgezahlt(_empfaenger, _betrag);
    }
```

```
    // funktion für die Einzahlung
    function einzahlen() public payable {
        require(msg.value > 0, "Ether wurde nicht gesendet.");
        kontostand[msg.sender] += msg.value;
        emit eingezahlt(msg.sender, msg.value);
    }
```

```
    receive() external payable {
        emit etherEmpfangen(msg.sender, msg.value);
    }


    // vordefinierte Funktion: wenn Funktion nicht existiert
    fallback() external payable {
        emit FallbackGenutzt(msg.sender, msg.value, msg.data);
    }
```

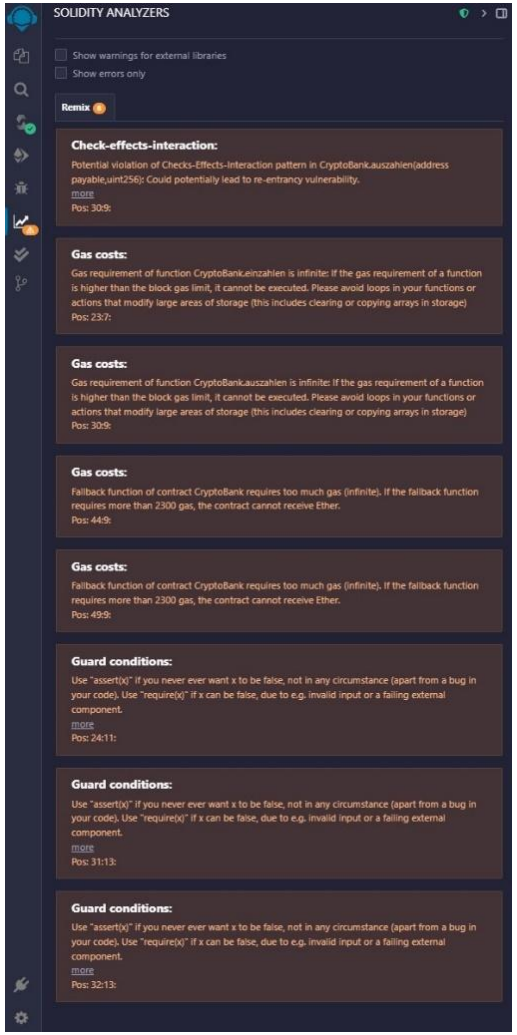
Development and Analysis (CryptoBank)

Remix Analyzer

- Check-Effects-Interaction (Reentrancy risk)
- Gas Costs
- Guard conditions

Slither Analyzer

- Reentrancy warning in auzahlen()
- Solidity version ^0.8.0
- Stylistic notice (events & parameter)
- Recommendation: set owner as immutable



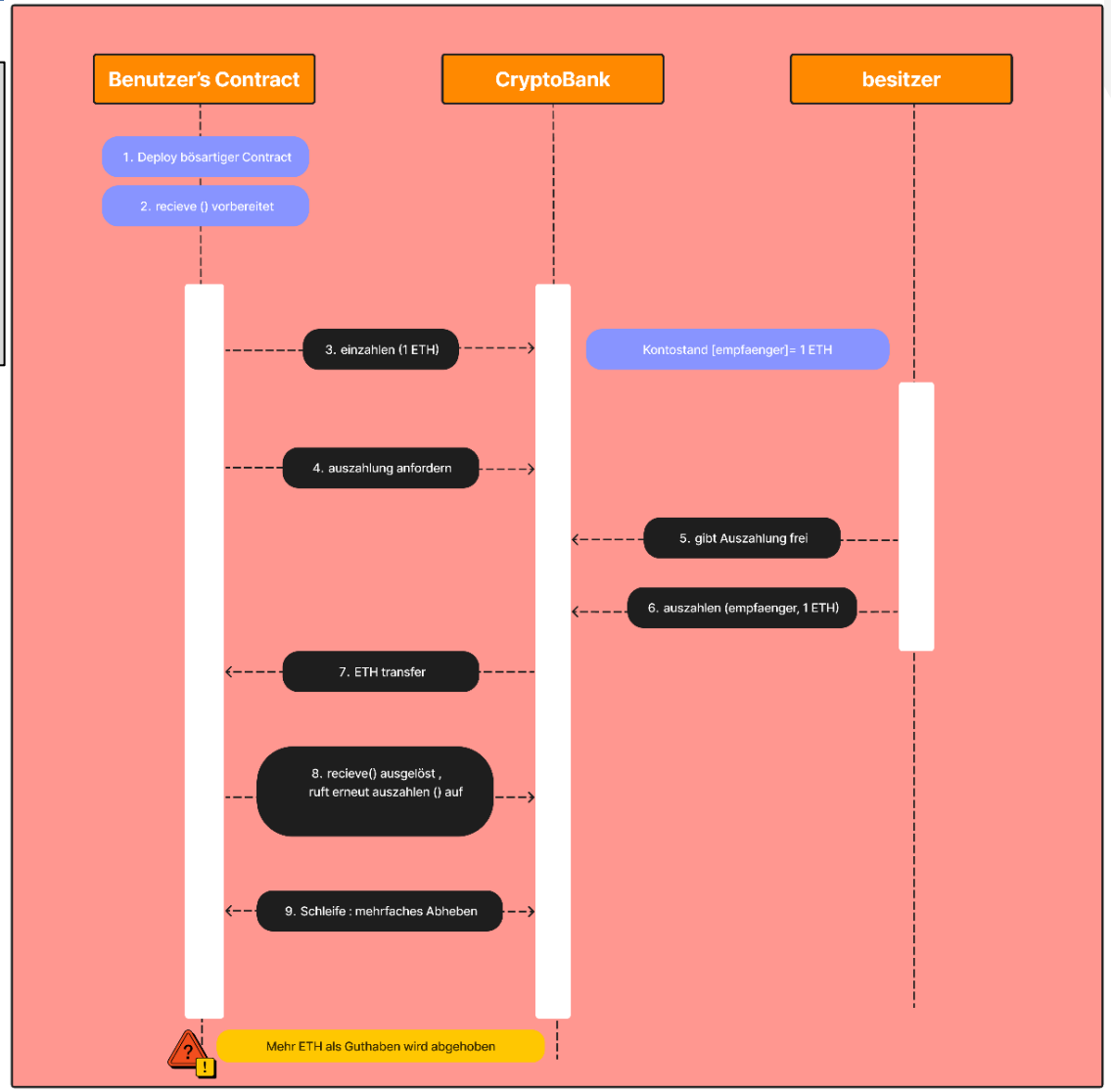
Remix Analyzer

```
PS C:\Users\Yousef\Desktop\Bachelor\Eigener Smart Contract> slither CryptoBank.sol
'solc --version' running
'solc CryptoBank.sol --combined-json abi,ast,bin,bin-runtime,srcmap,srcmap-runtime,userdoc,devdoc,hashes --allow-paths .,C:\Users\Yousef\Desktop\Bachelor\Eigener Smart Contract' running
INFO:Detectors:
Version constraint ^0.8.0 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess
        - AbiReencodingHeadOverflowWithStaticArrayCleanup
        - DirtyBytesArrayToStorage
        - DataLocationChangeInInternalOverride
        - NestedCalldataArrayAbiReencodingSizeValidation
        - SignedImmutables
        - ABIDecodeTwoDimensionalArrayMemory
        - KeccakCaching.
It is used by:
        - ^0.8.0 (CryptoBank.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Event CryptoBank.eingezahlt(address,uint256) (CryptoBank.sol#12) is not in CapWords
Event CryptoBank.ausgezahlt(address,uint256) (CryptoBank.sol#13) is not in CapWords
Event CryptoBank.etherEmpfangen(address,uint256) (CryptoBank.sol#14) is not in CapWords
Parameter CryptoBank.auszahlen(address,uint256)._empfaenger (CryptoBank.sol#30) is not in mixedCase
Parameter CryptoBank.auszahlen(address,uint256)._betrag (CryptoBank.sol#30) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in CryptoBank.auszahlen(address,uint256) (CryptoBank.sol#30-36):
        External calls:
        - _empfaenger.transfer(_betrag) (CryptoBank.sol#34)
        Event emitted after the call(s):
        - ausgezahlt(_empfaenger,_betrag) (CryptoBank.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
CryptoBank.besitzer (CryptoBank.sol#8) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:CryptoBank.sol analyzed (1 contracts with 100 detectors), 8 result(s) found
PS C:\Users\Yousef\Desktop\Bachelor\Eigener Smart Contract>
```

Slither

Development and Analysis (CryptoBank)

```
// funktion für die Auszahlung: Diese Funktion wird nur von Besitzer ausgeführt
function auszahlen(address payable _empfaenger, uint _betrag) public {
    require(msg.sender == besitzer, "Sie sind nicht der Besitzer!");
    require(kontostand[_empfaenger] >= _betrag, "Kontostand ist nicht genug!");
    kontostand[_empfaenger] -= _betrag;
    _empfaenger.transfer(_betrag);
    emit ausgezahlt(_empfaenger, _betrag);
}
```



Improvements and Re-Analysis (CryptoBankSecure)

```
// NEU: Importieren des Sicherheitsmodules für Reentrancy Schutz
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

// NEU: Erben von ReentrancyGuard
contract CryptoBankSecure is ReentrancyGuard {
```

```
address public immutable besitzer;
mapping(address => uint) public kontostand;
```

```
// geändert: Events in CamelCase für bessere Konventionen
event Eingezahlt(address indexed Benutzer, uint Betrag);
event Ausgezahlt(address indexed Empfaenger, uint Betrag);
event EtherEmpfangen(address Absender, uint Betrag);
// Fallback Event wurde entfert: Gas sparen
```

```
// Geändert: fallback() enthält nur revert und vermeidet Gaswarnung
fallback() external payable {
    revert("Unbekannte Funktion aufgerufen.");
}
```

```
// Geändert mit Reentrancy Schutz
function auszahlen(address payable empfaenger, uint betrag) public nonReentrant {
    require(msg.sender == besitzer, "Sie sind nicht der Besitzer!");
    require(kontostand[empfaenger] >= betrag, "Kontostand ist nicht genug!");

    kontostand[empfaenger] -= betrag;

    // Geändert mit Sicherer Transfer durch call + require
    (bool success, ) = empfaenger.call{value: betrag}("");
    require(success, "Transfer fehlgeschlagen.");

    emit Ausgezahlt(empfaenger, betrag);
}
```

Improvements and Re-Analysis (CryptoBankSecure)

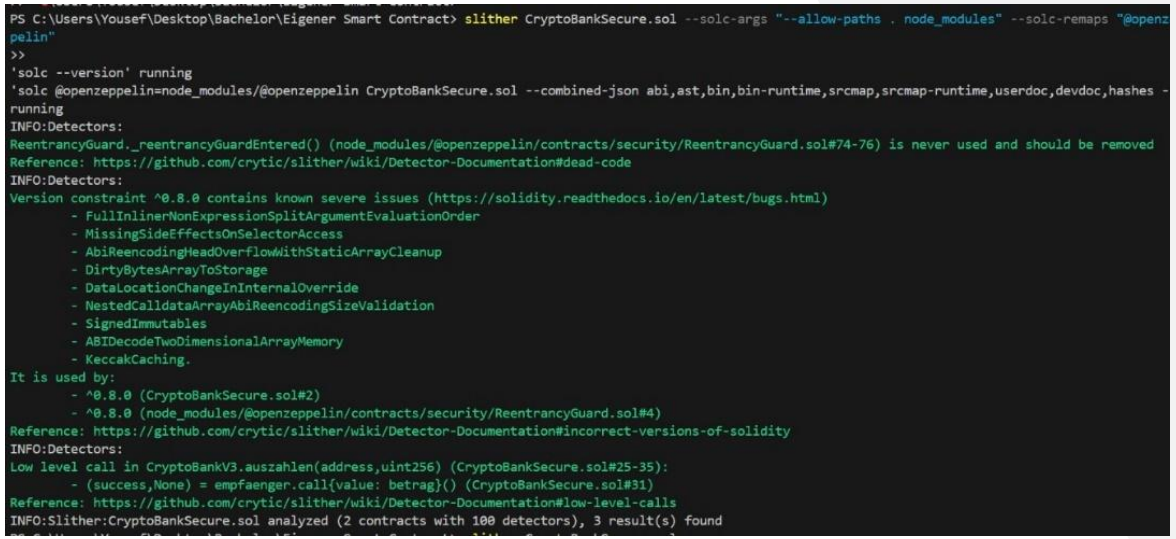
Remix Analyzer

- Check-Effects-Interactíon
- Low-Level-Call
- Guard Conditions
- Gas Costs

Slither Analyzer

- Low-Level-Call
- Solidity version ^0.8.0

Remix Analyzer



Slither

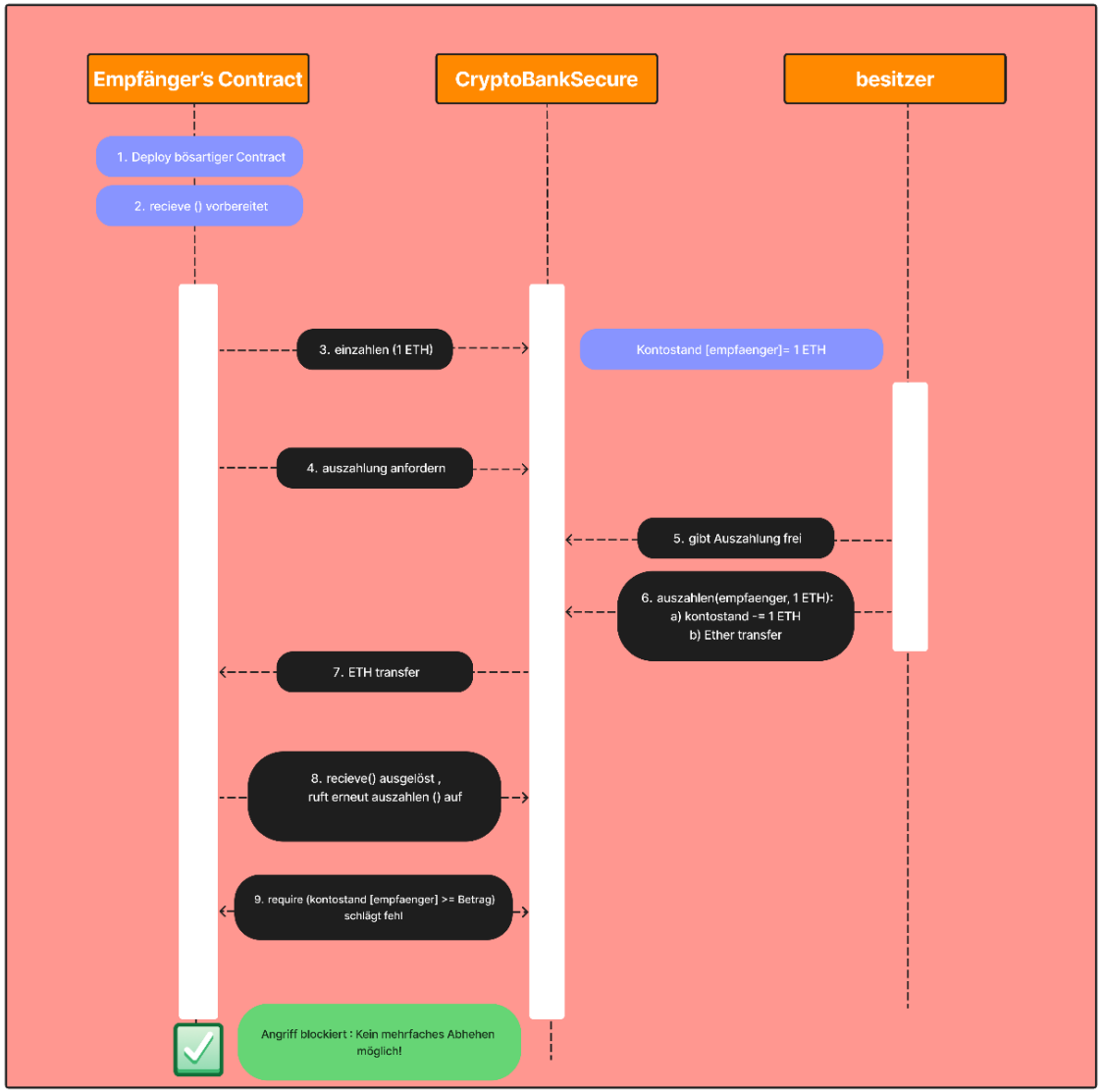
Improvements and Re-Analysis (CryptoBankSecure)

```
// Geändert mit Reentrancy Schutz
function auszahlen(address payable empfaenger, uint betrag) public nonReentrant {
    require(msg.sender == besitzer, "Sie sind nicht der Besitzer!");
    require(kontostand[empfaenger] >= betrag, "Kontostand ist nicht genug!");

    kontostand[empfaenger] -= betrag;

    // Geändert mit Sicherer Transfer durch call + require
    (bool success, ) = empfaenger.call{value: betrag}("");
    require(success, "Transfer fehlgeschlagen.");

    emit Ausgezahlt(empfaenger, betrag);
}
```



Results and Discussion

Analysepunkt	CryptoBank (ursprünglich)	CryptoBankSecure (optimiert)	Tool-Meldung	Behandlungsmaßnahme	Zweck / Verbesserung
Ether-Transfer	`transfer()` ohne Rückgabekontrolle	`call {value: ..}()+ require (success)`	Remix + Slither	Low-Level-Call mit Prüfung	Absicherung gegen fehlgeschlagene Auszahlungen
Reentrancy-Schutz	Kein Schutz vorhanden	`nonReentrant` verwendet (OpenZeppelin)	Remix (Warnung)	Modifier eingeführt	Verhinderung verschachtelter Wiederaufrufe
Fallback-Logik	Event + Speicherzugriff	nur `revert("..-")`	Remix	Logik entfernt	Minimierung von Gasverbrauch & Angriffspotenzial
Event-Namen	`eingezahlt`, `ausgezahlt`, `etherEmpfangen`	`Eingezahlt`, `Ausgezahlt`, `EtherEmpfangen`	Slither	Namensformat korrigiert	Stilistische Konformität mit Solidity Standards
Zugriffsprüfung	`require(msg-sender == besitzer)`	identisch	Remix	unverändert korrekt	Schutz vor unbefugtem Zugriff
besitzer-Variable	`public`, veränderbar	`public immutable`	Slither	auf `immutable` geändert	Effizienz & Absicht der Unveränderbarkeit klar dargestellt
require / assert	korrekt, aber Remix schlägt `assert()` vor	bei `require()` belassen	Remix	keine Änderung nötig	gemäß Dokumentation korrekt eingesetzt
Gasabschätzung	"infinite gas" bei mehreren Funktionen	weiterhin vorhanden	Remix	bekanntes Verhalten	keine tatsächliche Einschränkung
Compiler-Version	`^0.8.0` mit allgemeinen Warnungen	beibehalten	Slither	Version bewusst beibehalten	Stabilität und Kompatibilität mit Bibliotheken

Summary and Outlook

Summary

- Goal: Security analysis of Ethereum Smart Contracts
- Combined theory and practice (CryptoBank & CryptoBankSecure)
- Risks identified Remix & Slither
- Optimization: Reentrancy protection,secure Ether transfer, simplified fallback()
- Research question positively asnwered: risks identifiable and specifically fixable.

Outlook

- Blockchain gains importance beyond technology itself
- Smart Contracts: strong potential but also challenges.
- Challenges: scalability, law, usability
- Future: combination with AI, oracles and automated contract law.

References

Slide 5: **Jain, Shashank Mohan**. 2023. A Brief Introduction to Web3: Decentralized Web Fundamentals for App Development. Berkeley, CA : Apress, 2023.

Antonopoulos, Andreas M. und Wood, Gavin . 2019. Ethereum – Grundlagen und Programmierung: Smart Contracts und DApps entwickeln. Heidelberg : dpunkt.verlag, 2019.

Slide 7: **(1) Remix**. Remix IDE Documentation. [Online] https://remix-ide.readthedocs.io/ . **(2) Solidity**. Solidity Documentation. [Online] https://docs.soliditylang.org/en/v0.8.29/ . **(3) Remix Project**. Solidity Analyzers. [Online] https://remixide.readthedocs.io/en/latest/static_analysis.html . **(4) Trail of Bits**. Slither – Solidity Static Analysis Framework. [Online] https://github.com/crytic/slither.

Slide 10: **(1) Wood, Gavin,** Ethereum: A Secure Decentralised Generalised Transaction Ledger – Yellow Paper. [Online] https://ethereum.github.io/yellowpaper/paper.pdf. **(2) Ethereum Foundation**. Solidity – Security Considerations – Re-Entrancy. [Online] https://docs.soliditylang.org/en/latest/security-considerations.html#re-entrancy . **(3) ConsenSys Diligence.** Smart Contract Best Practices – Known Attacks: Reentrancy. [Online] https://consensys.github.io/smart-contract-best-practices/known_attacks/#reentrancy .

Slide 13: **(1) Solidity**. Security Considerations. [Online] https://docs.soliditylang.org/en/v0.8.30/securityconsiderations.html#re-entrancy . **(2) Solidity**. Control Structures. [Online] https://docs.soliditylang.org/en/v0.8.30/controlstructures.html#external-function-calls . **(3) Solidity**. Control Structures. [Online] https://docs.soliditylang.org/en/v0.8.30/controlstructures.html#error-handling-assert-require-revert-and-exceptions. **(4) Solidity**. Contracts. [Online] https://docs.soliditylang.org/en/v0.8.30/contracts.html#fallbackfunction . **(5)Trail of Bits**. Slither Usage Guide. [Online] https://github.com/crytic/slither/wiki/Usage#usingimports-from-a-package-manager. **(7) Trail of Bits**. Slither Detector Documentation. [Online] https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls.

Thank you very much for your interest in my Bachelor's thesis!