# Sicherheitsanalyse von Smart Contracts auf Ethereum

Bachelorarbeit im Studiengang Informatik (B.Sc.)

Frankfurt University of Applied Sciences – 2025

Yousef Ghanem

# Agenda

- Motivation, Zielsetzung und Forschungsfrage
- Methodik und Vorgehensweise
- Blockchain, Ethereum und Smart Contract
- Entwicklung und Analyse (CryptoBank)
- Verbesserungen und erneute Analyse (CryptoBankSecure)
- Ergebnisse und Diskussion
- Zusammenfassung und Ausblick

# Motivation, Zielsetzung und Forschungsfrage

## Motivation

- Hohe Relevanz in der Praxis
- Interesse an Sicherheit & Risiken
- Theorie & Praxis verbinden

## Zielsetzung

- Sicherheit von Smart Contracts auf Ethereum analysieren
- Typische Schwachstellen in der Praxis identifizieren

## Forschungsfrage

**„Welche sicherheitsrelevanten Risiken treten bei der Entwicklung von Smart Contracts auf Ethereum auf – und wie können diese wirksam erkannt und behoben werden?“**

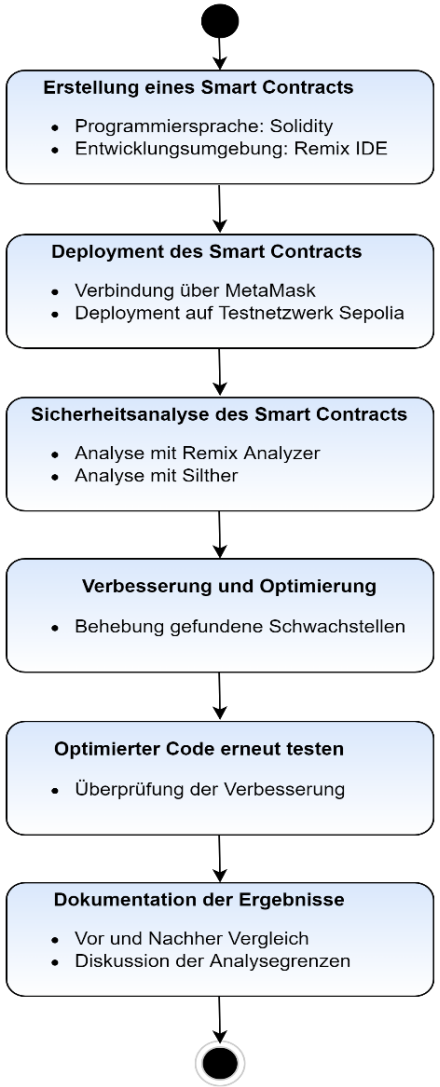
# Methodik und Vorgehensweise

## Theorie

- Blockchain, Ethereum und Smart Contracts

## Praxis

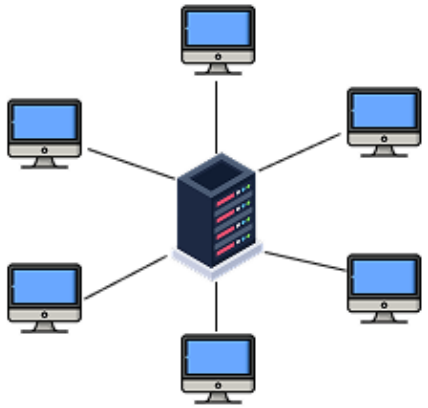
- Entwicklung mit Solidity & Remix
- Deployment auf Sepolia-Testnetz mit MetaMask
- Sicherheitsanalyse mit Remix und Slither
- Codeverbesserung & erneuter Test



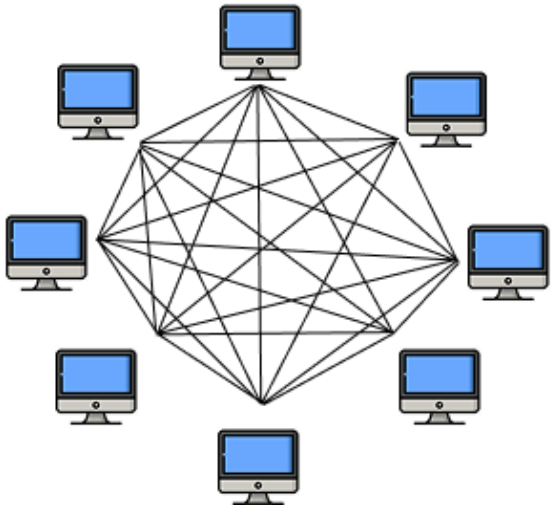
# Blockchain, Ethereum und Smart Contract

## Blockchain

- Dezentrale Datenstruktur aus verketteten Blöcken
- Manipulation kaum möglich



Zentralisiert



Dezentralisiert

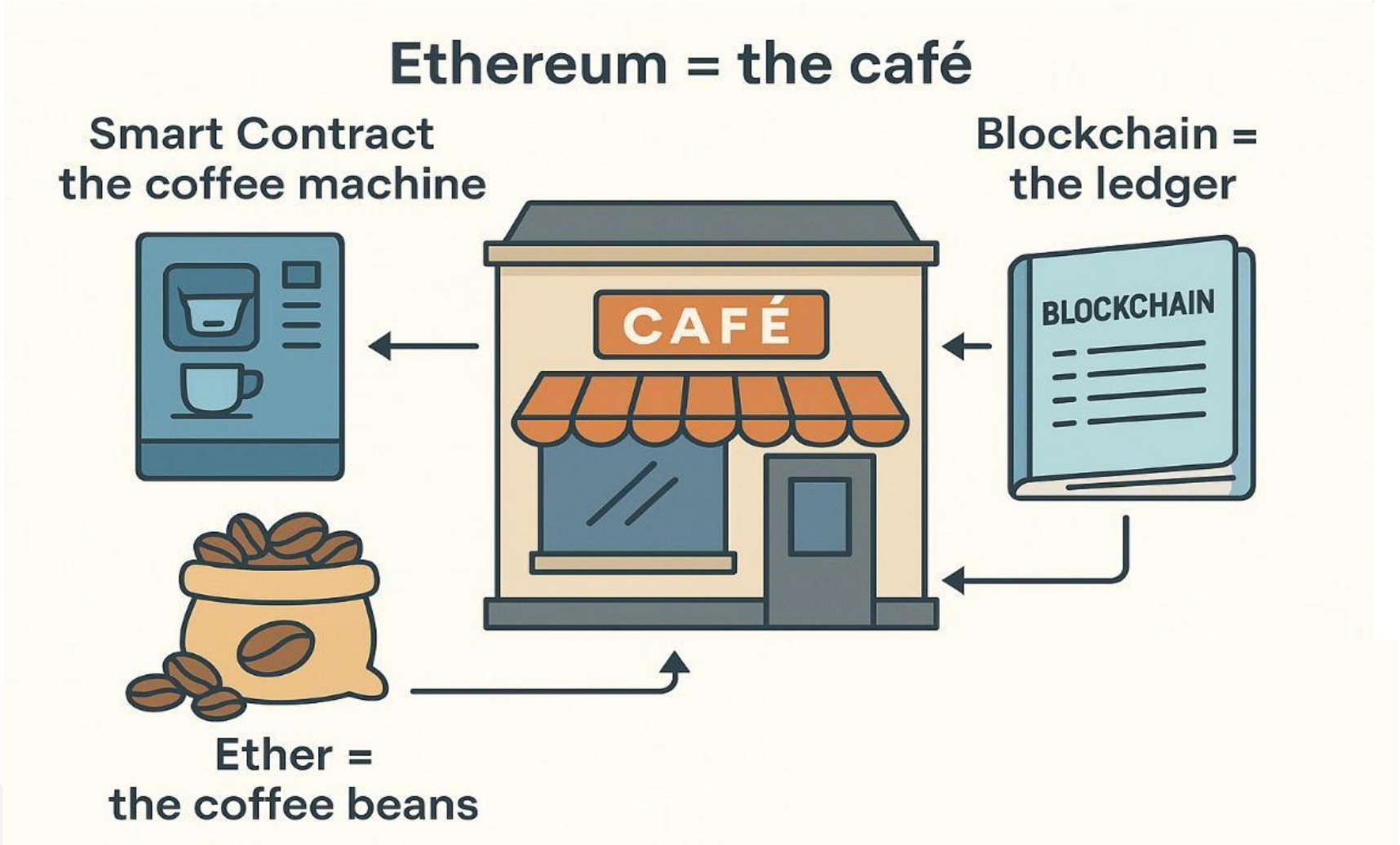
## Ethereum

- Plattform für dezentrale Anwendung
- Führt Programme in der Ethereum Virtual Machine aus

## Smart Contracts

- Automatisierte Verträge auf der Blockchain
- Unveränderlich & deterministisch

# Blockchain, Ethereum und Smart Contract



# Entwicklung und Analyse (CryptoBank)

## Entwicklung

- Remix IDE
  - Entwicklungsumgebung
  - Kompilieren & Deployen
- Solidity
  - Programmiersprache
  - Ähnlich wie JS & C++

## Deployen

- MetaMask
  - Wallet & Schnittstelle zum Ethereum Netzwerk
  - Verbinde Remix mit Sepolia
- Testnetz Sepolia
  - Offzielles Ethereum Testnetz
- Etherscan (Sepolia)
  - Blockchain-Explorer für Sepolia
  - Zeigt alles Transaktionen und Contracts

## Analyse

- Remix Analyzer
  - Analyse-Plugin
- Slither
  - Externes Analysetool (Terminal)
  - Detailierte Codeprüfung auf Sicherheitslücken

# Entwicklung und Analyse (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);
```

```
    // 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);
    }
```

```
    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);
    }
```

```
    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);
    }
```

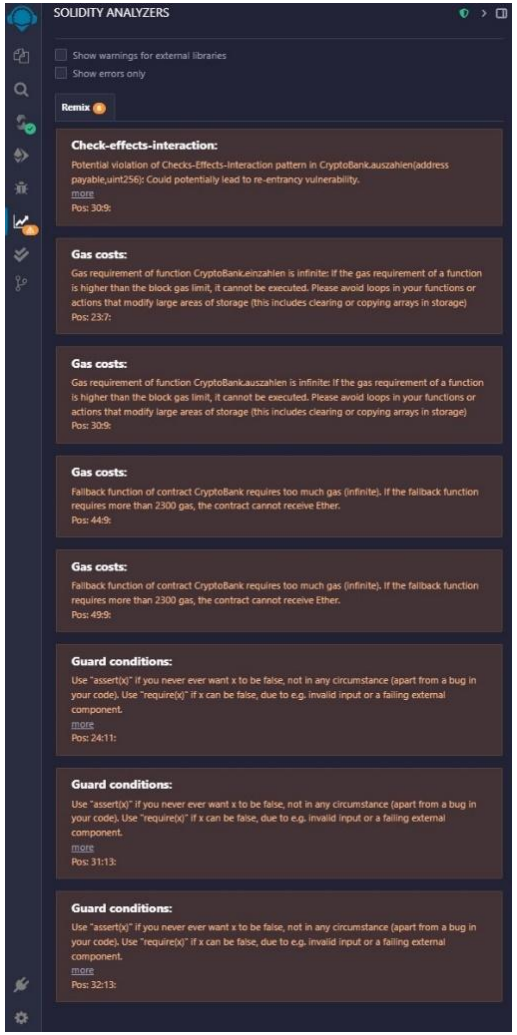
# Entwicklung und Analyse (CryptoBank)

## Remix Analyzer

- Check-Effects-Interactíon (Reentrancy-Risiko)
- Gasverbrauch
- Guard Conditions

## Slither Analyzer

- Reentrancy-Warnung in auzahlen()
- Solidity-Version ^0.8.0
- Stilistische Hinweis (Events & Parameter)
- Empfehlung: besitzer als immutable



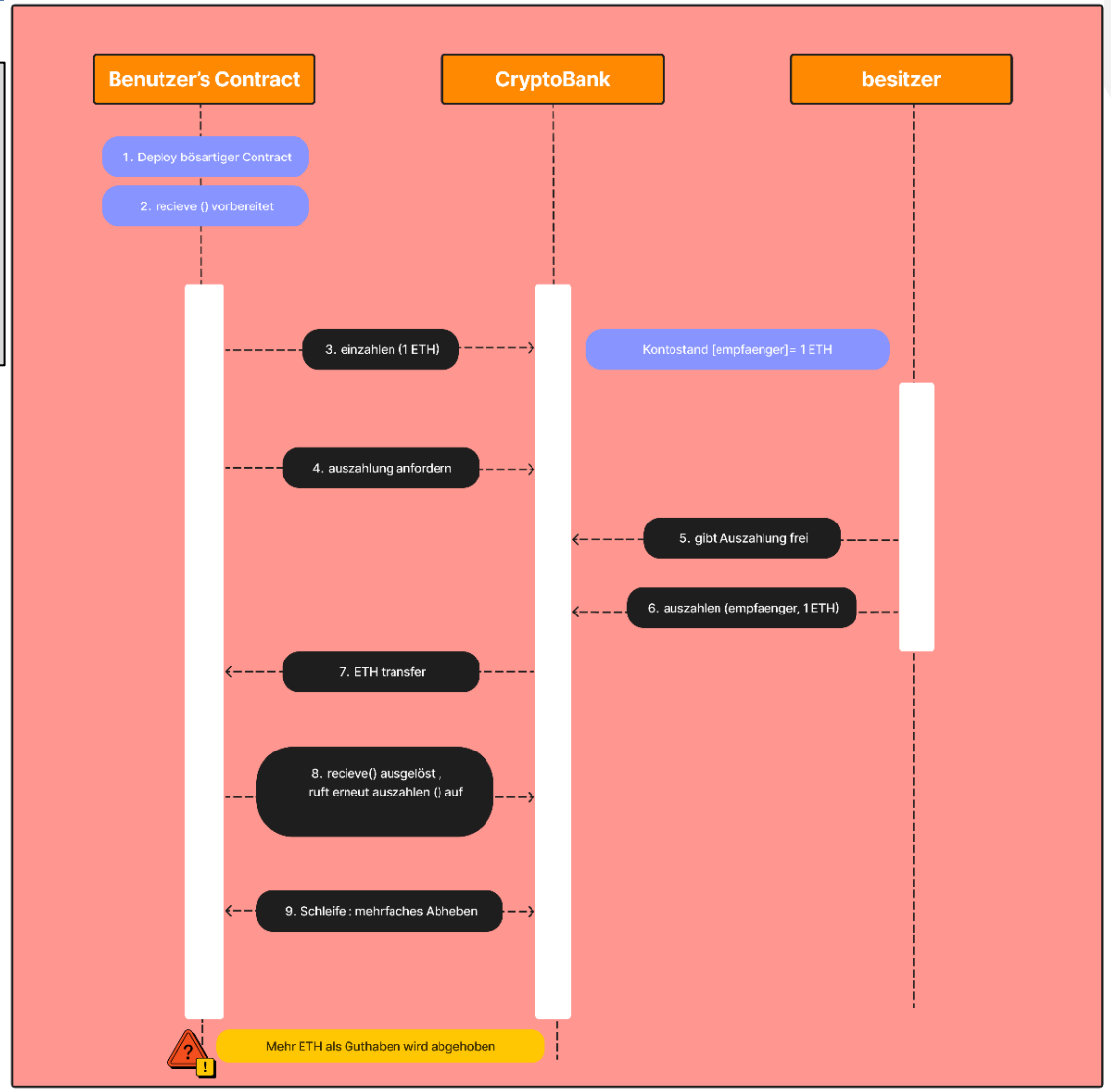

Remix Analyzer

```
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

# Entwicklung und Analyse (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);
}
```



# Verbesserungen und erneute Analyse (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);
}
```

# Verbesserungen und erneute Analyse (CryptoBankSecure)

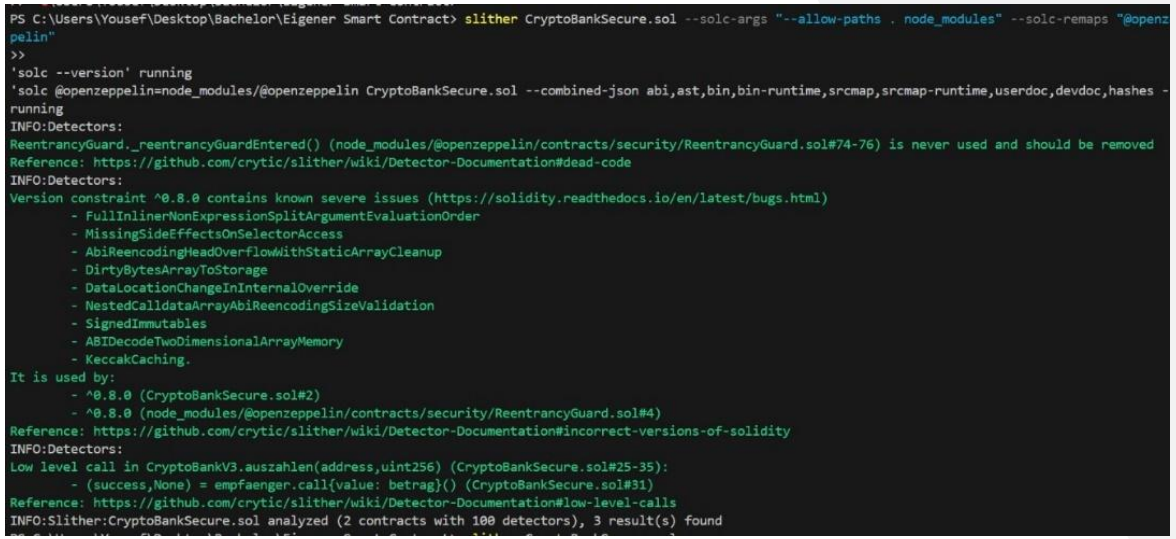
## Remix Analyzer

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

## Slither Analyzer

- Low-Level-Call
- Solidity-Version ^0.8.0


Remix Analyzer




Slither

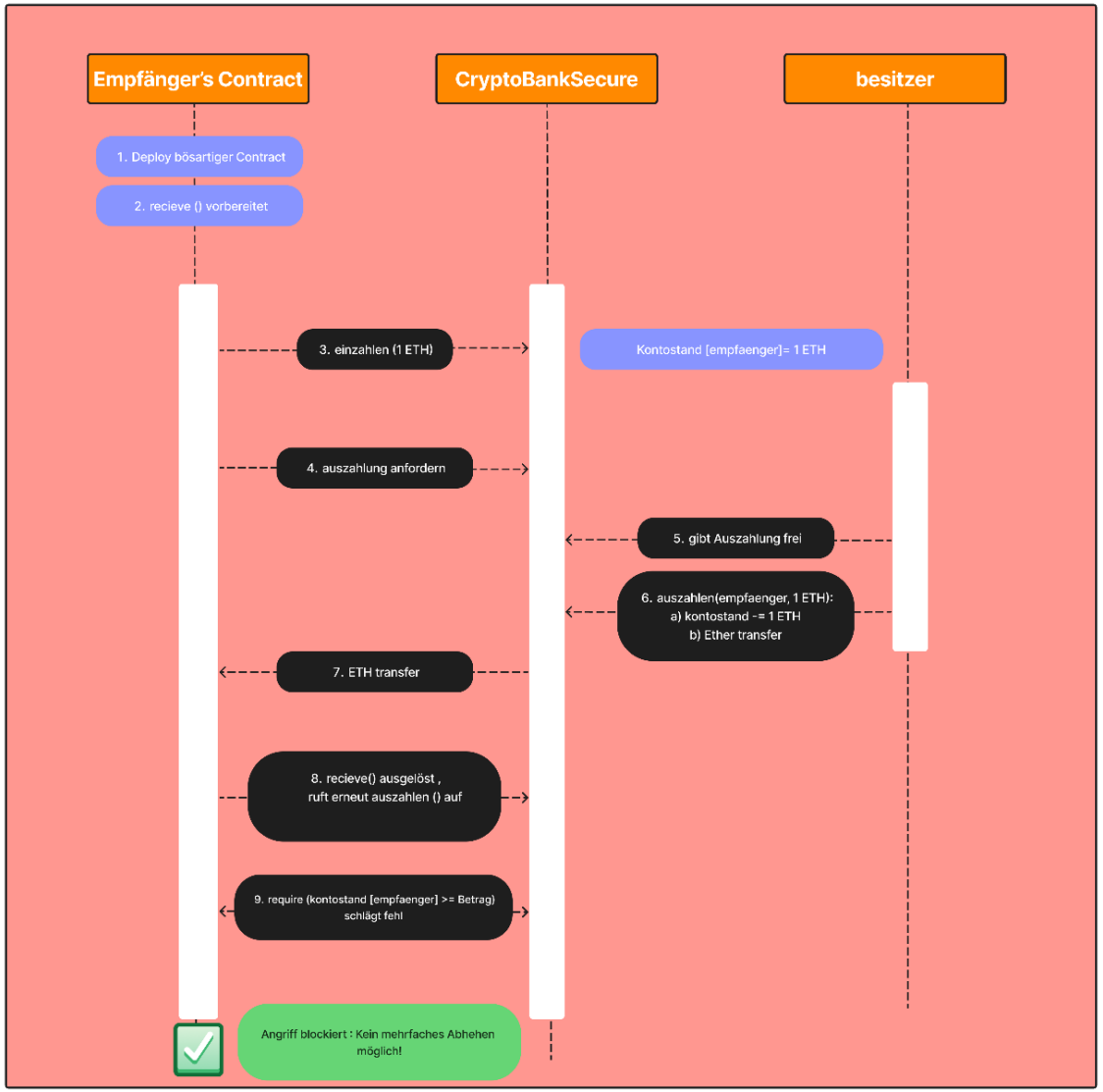
# Verbesserungen und erneute Analyse (CryptoBankSecure)

```
// 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);
}
```



# Ergebnisse und Diskussion

| 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 |

# Zusammenfassung und Ausblick

## Zusammenfassung

- Ziel: Sicherhietsanalyse von Ethereum Smart Contracts
- Theorie +Praxis kombiniert (CryptoBank & CryptoBankSecure)
- Risiken mit Remix & Slither erkannt
- Optimierung: Reentrancy-Schutz, sicherer Ether Transfer, vereinfachte fallback()
- Forschungsfrage positiv beantwortet: Risiken erkennbar und gezielt behebbar

## Ausblick

- Blockchain gewinnt an Bedeutung über Technik hinaus
- Smart Contracts: starkes Potenzial aber auch Hürden
- Herausforderungen: Skalierung, Recht, Bedienbarkeit
- Zukunft: Kombination mit KI, Oracles, automatischem Vertragsrecht

# Quellen


Folie 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.

Folie 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.

Folie 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 .

Folie 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.

# Vielen Dank fürs Interesse an meiner Bachelorarbeit!