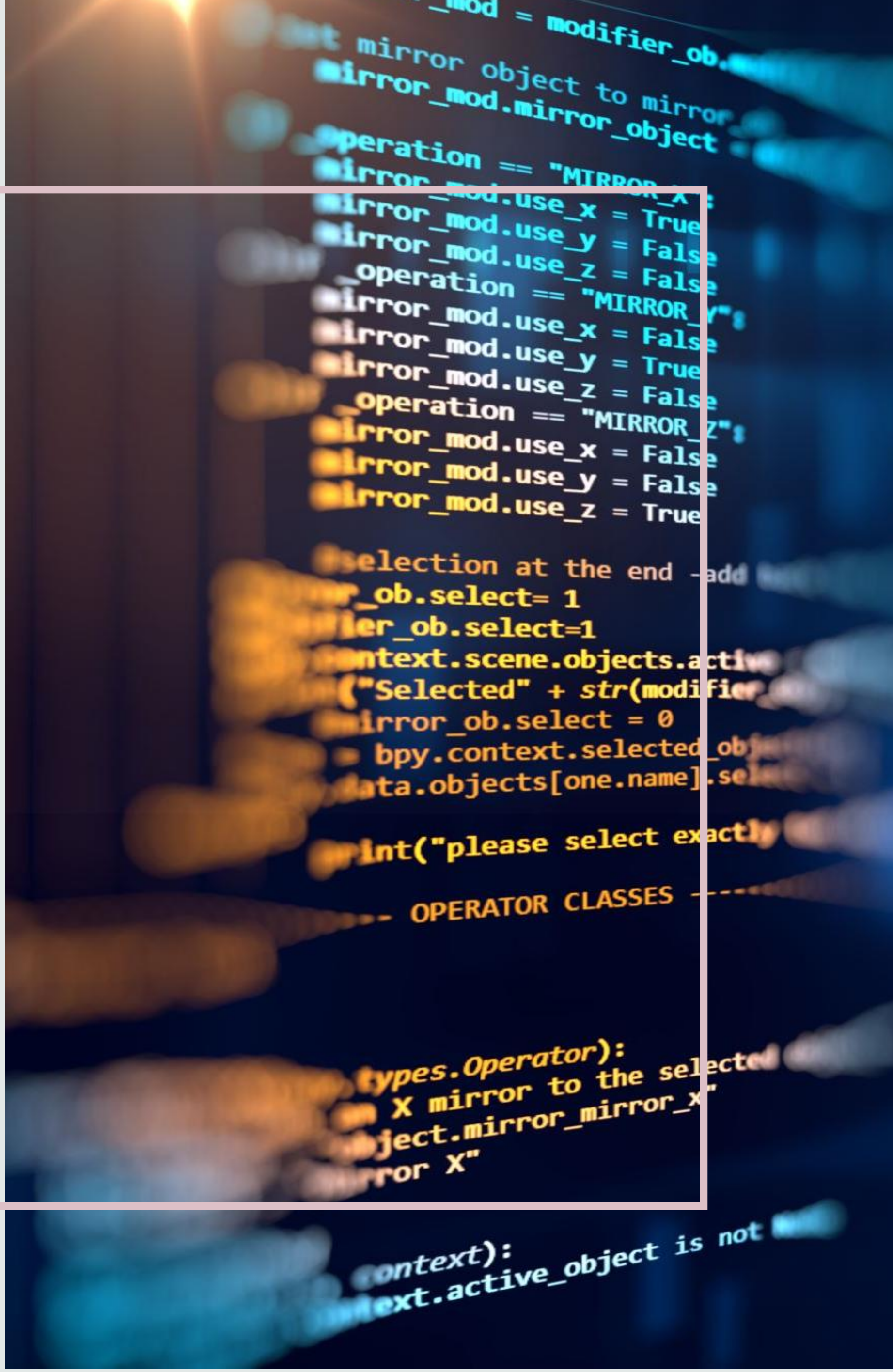
SOLIDITY COMPILER BUGS

Truong Do

WHAT IS SOLIDITY?

- Solidity is a high-level programming language that was designed for writing smart contracts on the blockchain networks. It is mainly used for developing decentralized applications (DApps) and smart contracts on blockchain platform like Ethereum.

  This allows developers to write code to implement smart contracts. Allowing them to interact with blockchain data and other contracts.

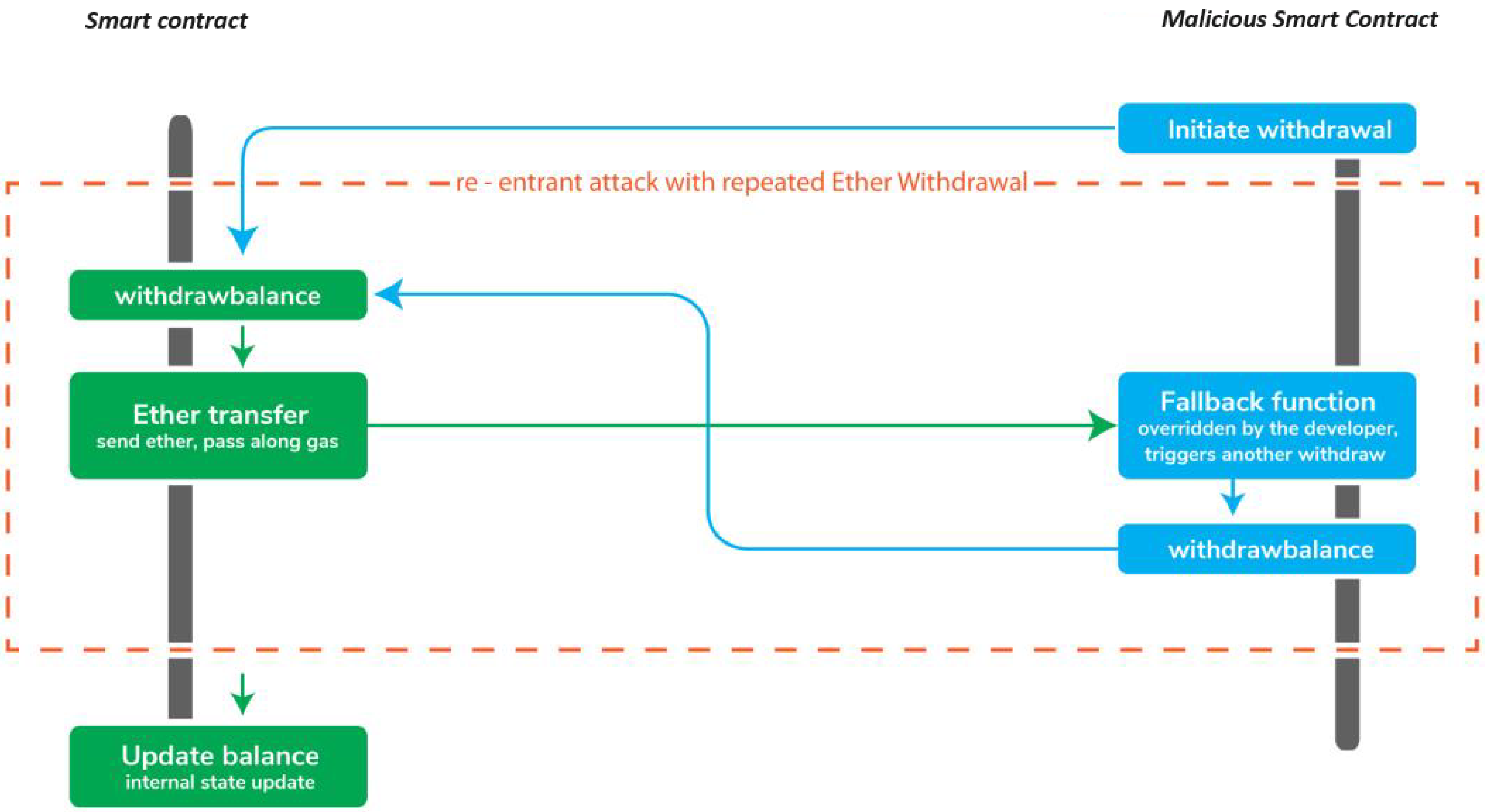


SOLIDITY COMPILER BUGS

- Solidity being the primary programming language for writing smart contracts are bound to have risks and various vulnerabilities.

REENTRANCY

- This occurs when a contract makes an external call to another contract or external function that allows the contract to revert to its original state before the first call was complete.
- When a smart contract makes an external call, the execution of Ethereum Virtual Machine (EVM) is transferred from calling a smart contract. As a result, there could be potential threat unless the original developer has a complete understanding of the smart contract's code.


Smart contract
Malicious Smart Contract
Initiate withdrawal
re - entrant attack with repeated Ether Withdrawal
withdrawbalance
Ether transfer
send ether, pass along gas
Fallback function
overridden by the developer, triggers another withdraw
withdrawbalance
Update balance
internal state update

FEI PROTOCOL

- In April 2022, the Fei Protocol fell victim of a reentrancy attack. The hacker took approximately $80 million in tokens.
- The hacker took advantage of two functions in the Fei protocol
- exitMarket – this function verifies that a deposit is no longer used as a collateral for a loan and then allows it to be withdrawn.
- Borrow – this function allows a user to take out a loan using a deposited asset as a collateral and does not follow the check-effect-interaction pattern

FEI PROTOCOL

- The attacker exploited this vulnerability by calling borrow, using a smart contract address.
- When the borrow function sends the loaned amount to the borrower, it has not yet been updated in its internal state to reflect that the deposited asset is currently being used as collateral
- The fallback function in the smart contract taking out the loan is executed when the loan is sent to it and calls the exitMarket function.

FEI PROTOCOL

- The hacker extract the deposit used as collateral for the loan
- This reentrancy vulnerability enables a hacker to take out a loan and also extract the deposited assets used as collateral for the loan
- By exploiting multiple different pools in the Fei Protocol smart contract. Hacker was able to drain 80$ million in tokens
- This attack was ranked #10 spot on Rekt’s leaderboard of DeFi hacks.

GAS LIMIT AND LOOPS

- Gas is a unit of measurement for the computational effort required to execute transactions on the Ethereum blockchain.
- Every transaction consumes a certain amount of gas, which is paid for by the sender of the transaction.
- If a transaction exceeds its specified gas limit during an execution. It will be automatically reverted and, any changes made to the state will be undone. However, the sender will still be charged for the failed transaction.

GAS LIMIT AND LOOPS

- Loops are fundamental programming construct that allow developers to execute a block of code repeatedly until a certain condition is met.
- Loops are related in the context of gas limit because they have the potential to consume a significant amount of gas if not used carefully.
- Infinite loops, where loop condition is always true and there is no code to exit the loop, are especially dangerous because they can lead to transactions running out of gas and results in loss of funds or unexpected behavior in the contract.

LOOPS SOLUTIONS

- Avoiding complex or nested loops that may consume excessive gas
- Using gas-efficient algorithms to minimize gas consumption
- Ensuring that loops have clear exit conditions to prevent infinite loops

Improper use of loops or excessive gas consumption can lead to denial of service (DoS) attacks or execution failures.