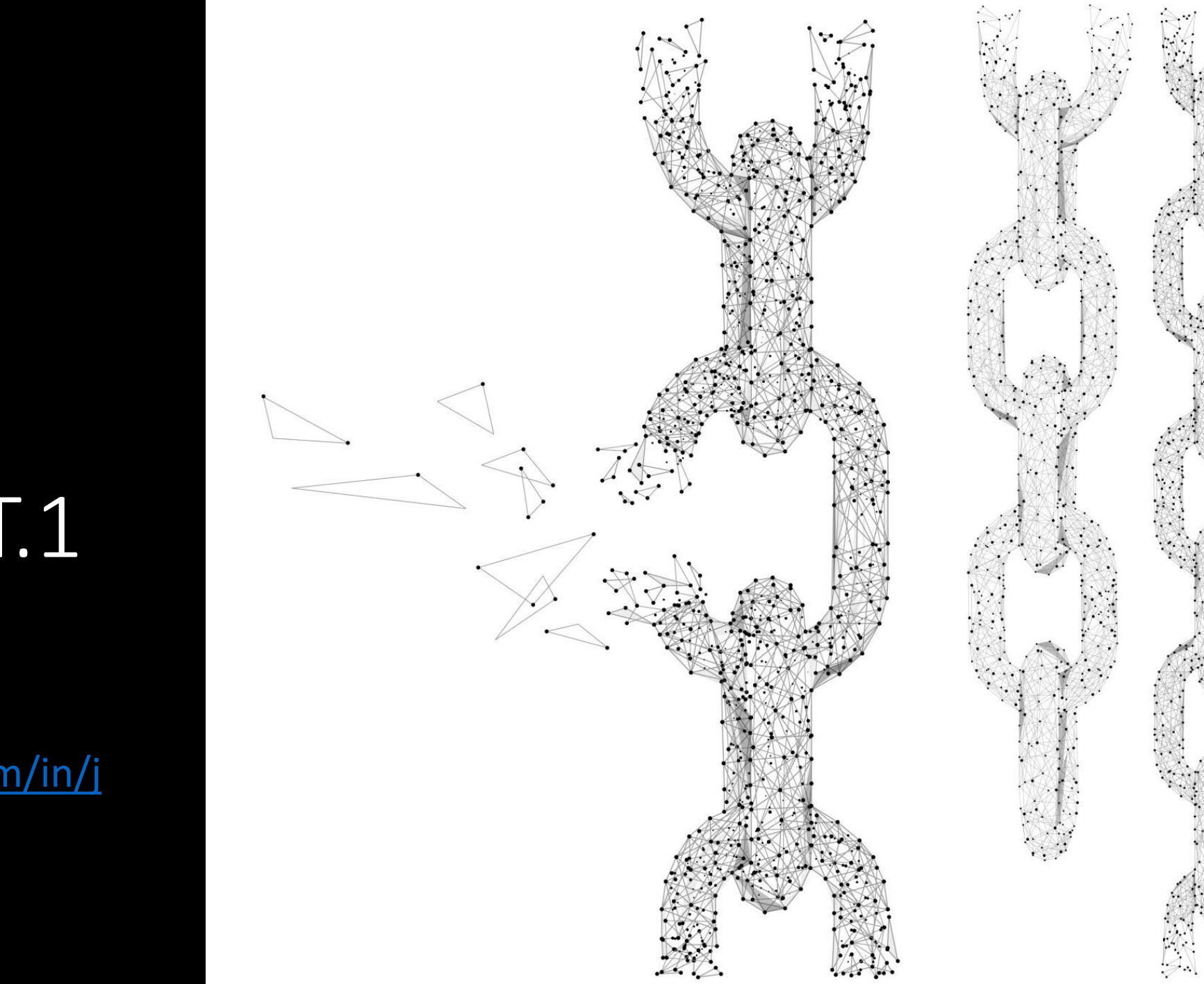
Smart Contract Security – Overview PT.1

Joas A Santos

https://www.linkedin.com/in/joas-antonio-dos-santos



Smart Contract Techniques

- Smart contract hacking techniques involve exploiting vulnerabilities in the code of a smart contract to gain unauthorized access or manipulate the contract's behavior. Here are some common smart contract hacking techniques:
- Reentrancy Attacks: This attack exploits a vulnerability that allows a malicious actor to re-enter the same function multiple times before the previous execution is complete. This can result in the malicious actor withdrawing more funds than they should be able to.
- Integer Overflow/Underflow: Smart contracts often use integer values to store data such as balances or timestamps. If the integer is not properly validated or has insufficient bounds checking, it can be manipulated by an attacker to overflow or underflow, resulting in unexpected behavior.

Smart Contract Techniques

- Unchecked External Calls: Smart contracts can call other contracts or external services. If the called contract is malicious or vulnerable, it can cause unexpected behavior in the calling contract.
- Denial of Service (DoS): A DoS attack can occur if an attacker exploits a vulnerability in a smart contract to consume resources such as gas or computational power, causing the contract to become unresponsive.
- Front-Running: Front-running involves exploiting the time delay between a transaction being submitted to the blockchain and its inclusion in a block. An attacker can place their own transaction in the blockchain to take advantage of information contained in another user's transaction.
- Malicious Libraries: Smart contracts may rely on external libraries or dependencies. If a malicious actor can modify or replace these libraries, they can introduce vulnerabilities or manipulate the behavior of the contract.

Cryptography in Blockchain

- Cryptography is a fundamental aspect of blockchain technology, providing the necessary security and privacy features that enable trust and authenticity in a decentralized network. Here are some key cryptographic techniques used in blockchain:

1. Hash Functions: Hash functions are used to convert input data of arbitrary length into fixed-size outputs, known as hashes. Hashes are used to identify and verify the integrity of data, as any change to the input data results in a different hash output. Blockchain uses hash functions to store transaction data and create a tamper-proof record of transactions.
2. Public Key Cryptography: Public key cryptography, also known as asymmetric cryptography, uses a pair of keys – a public key and a private key – to encrypt and decrypt data. Each user in the blockchain network has a public key that is shared with others, while the corresponding private key is kept secret. Public key cryptography is used to ensure secure communication and authentication in the blockchain network.
3. Digital Signatures: Digital signatures are used to provide authenticity and integrity to messages or transactions. A digital signature is generated using a private key and can be verified using the corresponding public key. In blockchain, digital signatures are used to prove ownership of assets and ensure that transactions are authorized by the correct parties.

Cryptography in Blockchain

Merkle Trees: Merkle Trees are used to efficiently store and verify large amounts of data in a blockchain. A Merkle Tree is a binary tree structure where each leaf node represents a hash of a piece of data, and each non-leaf node represents a hash of its children nodes. Merkle Trees allow for quick verification of data integrity without needing to store the entire blockchain.

Zero-Knowledge Proofs: Zero-knowledge proofs are cryptographic techniques that allow a user to prove knowledge of a secret value without revealing the value itself. This is useful in blockchain for providing privacy and anonymity, while still allowing for the validation of transactions or data.

Types of Blockchain and Different Blockchain technologies

- There are three main types of blockchain: public, private, and consortium. Each type has its own unique characteristics and use cases.

1. Public Blockchain: A public blockchain is a decentralized network where anyone can join and participate without any restrictions. Transactions are validated and recorded by a distributed network of nodes, and the blockchain is maintained by a community of users. Public blockchains are highly transparent, immutable, and censorship-resistant, making them well-suited for use cases such as cryptocurrencies and decentralized applications.

- Examples: Bitcoin, Ethereum, Litecoin, etc.

Types of Blockchain and Different Blockchain technologies

2. Private Blockchain: A private blockchain is a closed network where access is restricted to a select group of users or organizations. Transactions are validated and recorded by a pre-approved group of nodes, and the blockchain is maintained by a central authority or organization. Private blockchains offer more control, privacy, and scalability than public blockchains, making them well-suited for enterprise use cases.
   - Examples: Hyperledger Fabric, Corda, Quorum, etc.
3. Consortium Blockchain: A consortium blockchain is a hybrid of public and private blockchains. It is a decentralized network where multiple organizations join together to maintain the blockchain, but access is restricted to a pre-approved group of participants. Consortium blockchains offer the benefits of both public and private blockchains, such as transparency and security, while also allowing for more flexibility and scalability.
   - Examples: R3 Corda, IBM Blockchain Platform, etc.

Types of Blockchain and Different Blockchain technologies

- There are also several different blockchain technologies that are used in these different types of blockchains. Some of the most common blockchain technologies include:

1. Proof of Work (PoW): A consensus algorithm used in many public blockchains, where miners compete to solve complex mathematical puzzles to validate transactions and earn rewards.
2. Proof of Stake (PoS): A consensus algorithm where validators are selected based on the amount of cryptocurrency they hold, and the likelihood of being selected is proportional to their stake. PoS is more energy-efficient than PoW.
3. Delegated Proof of Stake (DPoS): A consensus algorithm where stakeholders vote on a group of delegates who are responsible for validating transactions and maintaining the blockchain.
4. Byzantine Fault Tolerance (BFT): A consensus algorithm used in many private and consortium blockchains, where a pre-approved group of validators must agree on the validity of transactions.
5. Directed Acyclic Graph (DAG): A data structure used in some blockchain technologies that allows for asynchronous transaction validation and scalability, without the need for miners or validators.

Solidity Language Programming

- Solidity is a high-level programming language used for writing smart contracts on the Ethereum blockchain. Smart contracts written in Solidity are compiled into bytecode, which can be executed on the Ethereum Virtual Machine (EVM).
- Here are some key features of Solidity smart contracts:

1. Object-Oriented Programming: Solidity supports object-oriented programming (OOP) concepts such as inheritance, polymorphism, and encapsulation. This allows for code reuse, modularity, and abstraction, making it easier to write and maintain complex smart contracts.
2. Built-In Data Types: Solidity supports various data types such as integers, booleans, strings, and arrays. It also supports custom data structures such as structs and mappings, which allow developers to define their own data types and store data in a structured way.
3. Event Logs: Solidity supports event logs, which are used to emit notifications when specific conditions are met in the smart contract. These events can be used to trigger external actions or provide feedback to users.
4. Ethereum Naming Service (ENS): Solidity supports the Ethereum Naming Service (ENS), which is a decentralized domain name system that allows users to register human-readable names for their Ethereum addresses. ENS names can be used in Solidity smart contracts, making it easier to create user-friendly dApps.
5. Smart Contract Standards: Solidity has several smart contract standards that are widely used on the Ethereum blockchain, such as ERC-20 (for fungible tokens), ERC-721 (for non-fungible tokens), and ERC-1155 (for both fungible and non-fungible tokens).

Smart Contract Vulnerabilities & Attacks

- https://pixelplex.io/blog/smart-contract-vulnerabilities/
- https://hacken.io/discover/smart-contract-vulnerabilities/
- https://blaize.tech/article-type/9-most-common-smart-contract-vulnerabilities-found-by-blaize/
- https://www.immunebytes.com/blog/smart-contract-vulnerabilities/
- https://github.com/kadenzipfel/smart-contract-vulnerabilities
- https://medium.com/coinmonks/smart-contracts-common-vulnerabilities-solidity-e64c5506b7f4

Smart Contract Vulnerabilities & Attacks

- Forcibly Sending Ether to a Smart Contract
- Insufficient Gas Griefing
- Reentrancy
- Integer Overflow and Underflow
- Timestamp Dependence
- Authorization Through tx.origin
- Floating Pragma
- Function Default Visibility
- Outdated Compiler Version
- Unchecked Call Return Value

Smart Contract Vulnerabilities & Attacks

- Unprotected Ether Withdrawal
- Unprotected Selfdestruct Instruction
- State Variable Default Visibility
- Uninitialized Storage Pointer
- Assert Violation
- Use of Deprecated Functions
- Delegatecall to Untrusted Callee
- Signature Malleability
- Incorrect Constructor Name
- Shadowing State Variables
- Weak Sources of Randomness from Chain Attributes
- Missing Protection against Signature Replay Attacks
- Requirement Validation

Smart Contract Vulnerabilities & Attacks

- Write to Arbitrary Storage Location
- Incorrect Inheritance Order
- Arbitrary Jump with Function Type Variable
- Presence of Unused Variables
- Unexpected Ether Balance
- Unencrypted Secrets
- Faulty Contract Detection
- Unclogged Blockchain Reliance
- Inadherence to Standards
- Unprotected Callback
- Asserting EOA from Code Size
- Transaction-Ordering Dependence
- DoS with Block Gas Limit
- DoS with (Unexpected) revert

Smart Contract and Solidity Hacking

- https://www.cobalt.io/blog/hacking-solidity-smart-contracts
- https://www.youtube.com/watch?v=TichhHxQ0zs&ab_channel=DappUniversity
- https://www.youtube.com/watch?v=XatbwCQ_HDY&ab_channel=DappUniversity
- https://www.youtube.com/watch?v=vcd6AoTf6Wk&ab_channel=DappUniversity
- https://medium.com/hackernoon/hackpedia-16-solidity-hacks-vulnerabilities-their-fixes-and-real-world-examples-f3210eba5148
- https://theblockchaintest.com/uploads/resources/RSA%20-%20Advanced%20Smart%20Contract%20Hacking%20-%202019.pdf
- https://www.hackread.com/smarter-smart-contracts-defi-hacks/
- https://hackernoon.com/how-to-hack-smart-contracts-self-destruct-and-solidity

Smart Contract and Solidity Hacking

- https://solidityscan.com/
- https://medium.com/hackernoon/scanning-ethereum-smart-contracts-for-vulnerabilities-b5caefd995df
- https://www.youtube.com/watch?v=B7sVGFc2G8A&ab_channel=SANSOffensiveOperations
- https://hackernoon.com/scanning-ethereum-smart-contracts-for-vulnerabilities-b5caefd995df
- https://makersden.io/blog/laser-cannon-hacking-smart-contracts
- https://smartcontractshacking.com/
- https://github.com/saeidshirazi/Awesome-Smart-Contract-Security

Blockchain PenTest

- https://medium.com/quillhash/blockchain-pentesting-penetration-testing-for-blockchain-networks-e37d8c325181
- https://vapt.ee/offensive-security/penetration-testing/emerging-technologies/blockchain-penetration-testing/
- https://audits.quillhash.com/services/blockchain-pen-testing
- https://www.arridae.com/services/Blockchain-Penetration-testing.php
- https://github.com/gokulsan/awesome-blockchain-security-platforms
- https://github.com/slowmistio/BlockChain-Security-List
- https://github.com/go-outside-labs/blockchain-hacking
- https://github.com/bobby-lin/study-blockchain
- https://github.com/brcyrr/PracticalCyberSecurityResources