Homework2

Part 1 (Security Attack Case Studies)

Read the following materials about security attack case studies, double spending of three 51% attack events in Bitcoin Gold, ZenCash (a fork of ZCash), and Verge:

1. Double spending attack in Bitcoin Gold (a fork of Bitcoin): https://forum.bitcoingold.org/t/double-spend-attacks-on-exchanges/1362

2. Double spending attack in ZenCash (a fork of ZCash): https://blog.zencash.com/zencash-statement-on-double-spend-attack/

3. 51% attack in Verge: https://www.anythingcrypto.com/guides/verge-network-mining-2018

4. Timestamp Hijacking Attack (Extra material to help understand 51% attack in Verge): https://cointelegraph.com/news/yale-research-proposes-factors-for-crypto-price-prediction

You may also want to search the web for additional materials. Then answer the following questions.

In the 51% events of Bitcoin Gold and ZenCash, the attacker actually manages to control more than half of computation powers for a short period of time. How could this be possible?

At the time where the attack launches, Bitcoin Gold and ZenCash share the same Proof-of-Work algorithm as ZCash (equihash with parameter N=200 and K=9). Do you think this is a good design for security? Why?

Nicehash is a website where people can pay BTC to rent computation power to solve Proof-of-Work. How does this affect the security of different cryptocurrecies (think about those cryptocurrecies that share the same PoW algorithm)?

ZenCash rightnow still uses the standard equihash (N=200, K=9). Suppose a double-spending attack can be launched if the attacker can control more than 50% of the computation power of the network **for one hour**. What is the cost for launching a double spending attack via renting PoW hashpower in Nicehash?

Verge uses multiple mining algorithm. Please explain the merged mining mechanism in Verge. How does it work?

The double spending attack in Verge depends first on a successful time hijacking to manipulate the difficulty of its Proof of Work algorithm. Why the time hijacking can influence the Proof of Work difficulty? Is this kind of manipulation possible in Bitcoin as well?

Any other questions you want to discuss in the class?

Part 2 (Ethereum)

Read the Ethereum white paper (https://github.com/ethereum/wiki/wiki/White-Paper). Answer the following questions:

Ethereum uses account model to store the blockchain state. One claims that the account model can reduce the average size of simple transfer transactions comparing to the UTXO model. Do you think this is true or not? Explain why.

One claims that comparing to the account model in Ethereum, the UTXO model can provide anonymous transactions if the user creates a new address for every transaction. Do you think this is true or not? Explain why.

Why Ethereum introduces a GAS limit for the block? What if we remove the GAS limit and put back the traditional block limit of 1MB like Bitcoin?

Ethereum sets up a different GAS amount for different EVM operations. Why?

Part 3 (Smart Contract Security Case Studies)

Ethereum enables users to deploy smart contracts, special programs that encode complicated transactions. Once deployed, those smart contracts are executed and enforced by all full nodes in the blockchain system. Unfortunately, smart contracts like any other programs may contain errors.

Read the following materials and answer questions:

1. Reentrance Vulnerability:
   https://hackernoon.com/smart-contract-security-part-1-reentrancy-attacks-ddb3b2429302
   https://vessenes.com/more-ethereum-attacks-race-to-empty-is-the-real-deal/

2. The DAO Hack:
   http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/

3. Integer Overflow Attack in BeautyChain:
   https://medium.com/@blockchain101/beautychain-erc20-integer-overflow-bug-explained-c583adcd847e
   https://medium.com/@peckshield/alert-new-batchoverflow-bug-in-multiple-erc20-smart-contracts-cve-2018-10299-511067db6536

The attacker in the DAO hack exploits a reentrance vulnerability in the DAO code. The attacker needs to write its own contract to exploit the vulnerability. What the fallback function of the attacker's contract should look like?

It is possible to put additional constraints on the fallback function implementation in the solidity language to prevent reentrance attack. For example, one could put a restrictive gas limit on the fallback function. What else you can think of?

What’s the consequence of the Integer Overflow attack in BeautyChain?

Search the web and collect other ERC20 tokens that suffered from integer overflow vulnerabilities.

Any other questions you want to discuss in the class?