CSC2125: Homework #2

Due on October 1

Fan Long

Thomas Hollis

Problem 1

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?

Solution

This is possible in a variety of different ways. Either the attacker increases his own share above 51% or he decreases the share of his peers below 49% (rare, only seen in small cryptocurrencies).

The first and most common possibility is that the attacker is already a big mining entity in the target coin and manages to increase his mining share temporarily by either purchasing more hash power or focussing his hash power on only one coin rather than spreading his mining power over multiple different coins.

Alternatively, the attacker could be the manager of a pool and could convince another pool to collude with him to launch a 51% attack that benefits both of them.

Another possibility is that the attacker is able to decrease the computation share of his peers by attacking the resources of larger users and pools. This is usually achieved via some direct attack for example: privilege execution / DDOS attacks on the main managing node of a large pool, etc.

In the case of other currencies like Verge, 51% attacks can be launched without necessarily controlling more than half of the computational power. This is often due to bugs which allow the attacker to hack a subsystem of the blockchain allowing himself to artificially reach the 51% control threshold (using timestamp hacking to artificially lower the difficulty level for example).

Problem 2

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?

Solution

This is usually poor security design because it makes 51% attacks slightly easier. This is because certain nodes can be bribed or hired (through Nicehash) to be purchased away from mining on one currency and instead concentrate their work on another. If the PoW hashing algorithm was different, these nodes would not be optimised to work on the new coin. This would make it harder for them to economically help achieve a 51% consensus. Therefore, sharing the exact same PoW could allow a malicious user to reach above the 51% threshold and launch an attack.

On the other hand, the fundamental golden rules of cryptography still apply. Do not roll out your own security if a good solution already exists. 256-bit AES dominates the symmetric encryption space for a good reason. Similarly, using well known and well tested hashing algorithms is better than trying to custom design your own. This could easily fail due to sloppy code, and could result in the new cryptocurrency being weakened or broken.

A good compromise would be to use known algorithms but with different parameters or slight variations to avoid being slightly more vulnerable to 51% attacks.

Problem 3

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 cryptocurrencies (think about those cryptocurrencies that share the same PoW algorithm)?

Solution

Nicehash can decrease the security of cryptocurrencies vulnerable to 51% attacks. This is because anyone can temporarily hire (for a short amount of time), a very large amount of hash power allowing themselves to go above the critical 51% threshold. While they will initially face costs associated with hiring this extra computational power, the return from a 51% attack can make it worth the cost at the expense of others (i.e. the victims of the double spending used during the 51% attack).

In addition, for cryptocurrencies that use the same PoW algorithm, this effect is even more pronounced as you can theoretically rent all the miners across all the currencies that share the same PoW algorithm and target them all on a single smaller cryptocurrency. If all cryptocurrencies had their own unique PoW algorithm this would be much more difficult to implement as you would not be able to convert a miner to start mining on a different protocol easily. This is because most of the mining power is concentrated through highly specialised ASICs which already have a part of the work pre-computed or optimised.

Problem 4

ZenCash right now 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 hash power in Nicehash?

Solution

The cost of a double spending attack achieved by renting through Nicehash is equal to the cost of renting a hash rate higher than the rest of the network combined. In the case of Zencash (now rebranded as horizen), the current network hash rate is 178.9 MH/s (24hrs average). Thus, the cost of renting a bit more than this on Nicehash is given by the following calculation (as of 26 Sep. 18):

Desired hash rate: $H_d > 178.9\text{MH/s}$
Hash rate price on Nicehash: $P = 0.0912$ BTC/(MSol/s)/day (minimum is 0.0733 BTC/(MSol/s)/day for equihash mining)
Unit conversion: 1 MSol/day $= 1'000'000$ H/day $=$ 1Mh/day.

Hence, the rental cost of a 51% attack for one hour is: $178.9 \times 0.0912/24 = 0.67982$ BTC $\approx$ \$4487.

It is worth noting however that Nicehash do not have the computational resources to achieve this by themselves as they have a cap on how much you can purchase through their marketplace (they will run out of hash power available to sell). The total equihash hashpower of Nicehash is currently around 100MH/s which would not be enough alone for a 51% attack. It is worth noting that since the market cap of Zencash is around \$50-100m, if a 51% attack was possible this easily everyone would be doing it. Indeed, ZenCash could however be attacked using Nicehash boosted by existing hardware to reach 51%.

Problem 5

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

Solution

Verge (previously known as DogecoinDark) uses multi-algorithm mining. This essentially means that miners can use various types of mining equipment to mine the Verge cryptocurrency because multiple different PoW hashing algorithms are supported. In the case of Verge, these supported hashing algorithms are X17, Lyra2rev2, myrgroestl, blake2s and Scrypt. All five of these algorithms have a 30s block target time and their difficulty is influenced only by the algorithm hash rate.

In fact the difficulty in Verge is calculated using the "Dark Gravity Wave". The details are out of the scope of this question but the difficulty is adjusted by block frequency looking at the blocks' timestamps. In fact, during the hack the blocks were timestamped roughly one hour before the present time so the difficulty crashed down from 1393093.39131 to 0.00024, as the network continuously estimated that the difficulty was too high as the block rate was seemingly very slow.

The consensus across the multiple algorithms works as follows. Each PoW mining algorithm is used during Verge mining cycles. Essentially, if a block is mined using algorithm A, a given amount of time must elapse until mining algorithm A is allowed again. This is achieved via timestamping. More specifically each submitted block is sent with the hash, the timestamp and the hashing algorithm used. This is how all 5 PoW hashing algorithms are each cycled through one after another, allowing for multiple mining consensus.

Merged mining on the other hand is a different concept. Merged mining is the process of permitting two different cryptocurrencies to be mined using the same hashes (i.e. both coins can accept the same valid hashes). This is usually implemented to boost the hashing power behind the blockchain networks of these coins. The best-known example of this is Namecoin and Bitcoin which can be merged mined. Indeed, Namecoin forked the SHA256 method employed by Bitcoin and thus has led to merged mining pools working on solving Bitcoin/Namecoin hashes simultaneously.

Verge does not support merged mining as of 26 September 2018 (after I discussed this with the TA, this turned out to be a typo and the question was modified but I thought I would include the information here anyway as it is interesting nonetheless).

Problem 6

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?

Solution

The 51% double spending attack in Verge was achieved by timestamp hijacking. As detailed before, Verge cycles through all 5 of its PoW mining algorithms. Therefore if a malicious user, named Trudy for example, was to successfully falsify her PoW timestamps, she could attack the Verge network. This is because Trudy would be able to submit blocks with fake timestamps allowing her to use the same mining algorithm repeatedly. This in turn means than when Trudy submits her spoofed blocks, she can outperform honest miners who have switched to a different algorithm. Essentially, Trudy would control a disproportionate amount of the hash power allowing her to carry out a 51% attack without having 51% of the mining power. It is

Problem 6 continued on next page. . .

interesting to note that when this happened, Verge introduced a hard fork to fix the vulnerability. A couple of months later, Verge was hit with yet another attack, this time a DDOS, which forced them to prepare another subsequent hard fork.

This kind of manipulation is therefore not possible in Bitcoin. This is mostly because Bitcoin only uses a single PoW hashing algorithm. It is however worth noting that timestamp manipulation is also possible in Bitcoin. However, the risk posed by this timestamp hacking is much less severe in Bitcoin than it is in Verge. Indeed, Bitcoin timestamping is achieved using a global network clock that is not perfectly in sync with real time. The network clock is adjusted based on the median of the reported clocks of peers in the network. It therefore has a minor fluctuation from time to time and is only roughly accurate. Thus far, no timestamp hacking attacks were successful in Bitcoin as far as we know.

Problem 7

1.7 Any other questions you want to discuss in the class?

Solution

- Have there been any physical 51% attacks whereby the attacker physically cuts the power to a few large mining farms or takes down a mining pool to allow himself to reach the 51% threshold?

- Are there protocols that involve more than majority to pass transactions (i.e. a 75% majority required)?

Problem 8

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.

Solution

I think that the statement the account model can reduce the average size of simple transfer transactions comparing to the UTXO model is true.

This is because of the nature of the UTXO model, in particular its multi-input/multi-output structure. This can be best explained by the following example:

Suppose a common merchant is using a blockchain with the UTXO model (like Bitcoin). This merchant receives, over the course of one month, many small transactions of funds. He will therefore generate a new address for every new transaction received, in order to maintain his anonymity. This means the merchant will end up with many UTXO with small amounts of BTC under his control. Whenever this merchant wants to spent however, he cannot use a single UTXO to pay as this is usually not enough to cover his larger transaction. The merchant will therefore use many UTXO as input to the transaction in order to be able to pay his larger sum of money. This leads to each transaction requiring a large number of redundant addresses, many more than required. Indeed, many transactions are like this whereby a large amount of addresses are used to pay a large sum which leads to larger average size of transactions.

Problem 8 continued on next page. . .

This is however not the case for the account model (such as the one in Ethereum). Indeed, in the account model only the address being debited and the address being credited are really required.

In fact, the best piece of evidence to support my belief is the fact that the average Bitcoin transaction is currently slightly larger than the average Ethereum transaction (despite the overheads of Ethereum).

Problem 9

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.

Solution

In my opinion the UTXO model, like the one implemented in BTC, cannot be fully anonymous even if a user creates a new address for every transaction. It is for this reason that Bitcoin is described as being pseudo-anonymous. The reason behind this is because UTXO means that all transactions have inputs and outputs and the blockchain is public. Hence the money can always be traced back. More specifically this is because of multi-input transactions. Even if you generate a new address for every incoming payment (which is by the way often inconvenient), as soon as you want to send a large payment, the wallet may pull bitcoins from multiple addresses. This links a group of addresses together as belonging to the same wallet. Hence your anonymity is only protected by nobody knowing that you are the owner of a particular wallet or collection of addresses (which is essentially impossible when you ask people to pay you).

However, this privacy issue can be improved by using multiple different wallets. In fact, services like MultiBit allow the management of multiple wallets within the same program. Similarly, a variety of different mixing services exist which will take your BTC and mix them with other users BTC and send them back out to you in multiple transactions. Essentially this makes tracking much harder but this relies on trusting the 3rd party with your funds and trusting this 3rd party not to keep logs. This is therefore not a good solution either.

Therefore, in my opinion, the UTXO model cannot provide truly anonymous transactions in a practical sense (many inconvenient) even if it can provide pseudo-anonymous ones to a pretty strong level. In my opinion, I would still rather use privacy focussed alternative currencies rather than BTC if privacy was my number one concern. Fortunately, these alternatives are numerous such as Zcash, Monero (a popular currency for purchasing illegal goods and services), Dash (or its PIVX fork), as well as many others.

Problem 10

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?

Solution

The GAS limit is introduced on blocks to substitute for the block limit present in Bitcoin. The aim is to keep transaction blocks at reasonable sizes (both in capacity and required computation). The fixed block size limit in Bitcoin is very inflexible and naive compared to the smarter custom GAS limit of Ethereum. In

Problem 10 continued on next page. . .

fact, the fixed 1MB block size of Bitcoin is a widely documented and hotly debated issue (since block size fundamentally limits the transaction speed). Ethereum's GAS limit is far better as it allows for a flexible blocksize limit, as voted on by miners. Indeed, this bypasses Bitcoin's main block size controversy and allows for stronger transaction speeds.

If we remove the GAS limit and put back the traditional block limit we lose the advantages of GAS over fixed block sizes. In addition, we are no longer able to use Ethereum for smart contracts as we can no longer measure the work needed by the network. This would render Ethereum basically useless as it can no longer achieve its primary purpose: smart-contracts/programming on the blockchain.

Problem 11

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

Solution

Ethereum introduced the flexible GAS amounts for a variety of reasons. GAS is the measure of computational effort (for computation, bandwidth and storage) required for one Ethereum client to send a transaction in the Ethereum blockchain network (paid for in ETH as a function of current GAS price). This limit is directly mapped to the operations neededed to be performed in the Ethereum Virtual Machine (EVM) to process that transaction. This was introduced to avoid misuse of the network by greedy users. Simply put each user pays for the EVM operations required to run the smart contracts needed.

EVM code has different GAS amounts for different EVM operations. This is because each operation requires a different amount of work. Hence it makes sense to have the amount of GAS for each operation different. The fee schedule is actually a tuple of 31 scalar values each corresponding to the cost in gas of abstract operation impacted by transactions. This is intuitive as certain operations may be very costly (computationally expensive) and should not be charged at the same rate as basic functions.

Indeed, Solidity and its corresponding compiled EVM code was designed to be Turing-complete. For this, it had to include loops and so it had to find a way to deal with loops that may run for ever (infinite loops), eating up all the resources of the network. GAS was the solution to this and infinite loops are not possible in ETH as they simply run out of gas and their execution is nullified and reverted (as per the ETH protocol).

Problem 12

The attacker in the DAO hack exploits a re-entrance vulnerability in the DAO code. The attacker needs to write its own contract to exploit the vulnerability. What the fallback function of the attackers contract should look like?

Solution

The fallback function of the attackers contract should look something like this:

Problem 12 continued on next page. . .

```
Function () public payable {
    if(limit >0){
        limit --;
        vul.withdrawEquity();
    }
}
```

This hack works because this will allow this function to recursively call the withdrawal function which in turn will call this function again until limit reaches 0. The limit and the if loop is added to prevent an error when withdrawing funds that are empty (any error would reverse the transaction thus reverse the hack).

Problem 13

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

Solution

It is indeed possible to place additional constraints in the fallback function implementation in Solidity. Interesting advice about fallback functions is given to us by Vitalik Buterin.

He explains fallback functions can be misused or abused thus we should, by convention, generally not use fallback functions except in very specific exceptions. For example. he also recommends that a restrictive gas limit should indeed be placed and increased to 25k GAS from the original 21k GAS (at the HLL level) for `send()` functions. In fact this function is currently safe against re-entrancy thanks to the gas limit, unlike the `call()` function.

Another additional constraint that I can think of is to add new conventions in the Solidity language and associated compilers. Perhaps this convention is established by compiler warnings or errors. This convention, could be recommending the use of full gas forwarding rather than using the `send()` command.

In addition, new libraries like the SafeMath library could be written to make the usage of remediation approaches easier. Indeed, two such approaches based on reordering race-to-empty functions or using mutexes have been suggested to help bypass the issue caused by the `send()` function.

Finally, a consensus page could be published on Github or along with documentation to make it clear what the potential risks are of using a function that relies on external code. This in fact has been done and is available at: `https://consensys.github.io/smart-contract-best-practices/`.

Problem 14

Whats the consequence of the Integer Overflow attack in BeautyChain?

Solution

This allowed anyone to create any number of BeautyChain tokens for free. In fact, around one vigintillion (1×10^{63}) BEC tokens were created. This in turn led to a devaluation of the token and its delisting from all

Problem 14 continued on next page. . .

major cryptocurrency exchanges.

However, a more long-term consequence of the integer overflow attack in BEC was that developers became increasingly careful about failing to use the SafeMath Solidity library. In addition, the CyberMiles compiler was developed to put in place some important safeguards to enforce the use of SafeMath in certain token contracts (ERC20, ERC223, ERC721, ERC884).

Problem 15

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

Solution

The tokens affected are (amongst others):
- UGToken (UGT)
- SmartBillions (SMART)
- BlockMesh/RightMesh/Mesh (MESH/MSH/MTC)
- SmartMesh (SMT)
- FirstCoin (FRST)
- GG Token (GG)
- CNY Token (CNY) & CNYTokenPLus (CNYt+)

Problem 16

Any other questions you want to discuss in the class?

Solution

- Is there any substitute for The DAO that is currently being worked on that has serious potential as a decentralised investment fund?
- Is anyone else aware that Satoshi Nakamoto initially proposed that the longest chain (i.e. the chain with the most blocks) is the one to follow but eventually decided it should be the strongest chain (i.e. the chain on which the most work has been done). This is because as difficulty changes so does the work done required to make a new block. Verge used longest chain unlike most other PoW currencies and this caused their hack. PoS currencies mostly still use longest chain as this is because it makes sense for PoS to use longest chain since no work is done. Why don't all new currencies opt for PoS with minting? What is the appeal of having PoW with mining?