A Brief Introduction to Blockchain Concepts

Howard L Poston

Published by GhostVolt

While blockchain technology has only become synonymous with recent methods in financial assets management it's applications are largely unlimited. But redistributing accountability for information into this new democracy poses new problems and practicalities, especially for the security conscious.

In this short edition we will explore the underlying concepts of blockchain technology, practical elements of what makes a blockchain and look at the inherent vulnerabilities of a decentralized network in the real world.

Author **Howard L Poston** is a freelance blockchain and cyber security trainer, consultant, and technical content developer.

Fundamentals

At the most basic level, blockchain technology is composed of cryptographic algorithms. The creator of blockchain, Satoshi Nakamoto, developed a system in which the trust that we traditionally place in organizations to maintain trusted records (like banks) is transferred to the blockchain and the cryptographic algorithms that it uses.

The Cryptography Behind Blockchain

The goal of the blockchain is to create a distributed, decentralized, and trusted record of the history of the system. The most famous blockchain, Bitcoin, uses this record to store the history of transactions, so people can make and receive payments on the Bitcoin blockchain and trust that their money won't be lost or stolen.

In order to achieve this level of trust, the blockchain uses a couple of cryptographic algorithms as building blocks. Hash functions and public key cryptography are crucial to both the functionality and security of the blockchain ecosystem.

Hash Functions

A hash function is a mathematical function that can take any number as an input and produces an output in a fixed range of numbers. For example, 256-bit hash functions (which are commonly used in blockchain), produce outputs in the range 0-2256.

In order to be considered secure, a hash function needs to be collision-resistant, this means that it's extremely difficult (to the point of being nearly impossible) to find two inputs that create the same hash output. Accomplishing this requires a few different features:

- No weaknesses in the hash function
- A large number of possible outputs
- A one-way hash function (can't derive the input from the output)
- Similar inputs produce very different outputs

If a hash function meets these requirements, it can be used in blockchain. However, if any of these requirements are violated, then the security of the blockchain is at risk. Blockchain relies heavily on secure hash functions to ensure that transactions cannot be modified after being stored in the ledger.

Public Key Cryptography

The other cryptographic algorithm used in blockchain technology is public key cryptography. This type of cryptography is also widely used on the Internet as well since it has so many useful properties. With public key cryptography, you can:

- Encrypt a message so that only the intended recipient can read it
- Generate a digital signature proving that you sent a given message
- Use a digital signature to verify that a message was not modified in transit

In public key cryptography, everyone has two different encryption keys: a private one and a public one. Your private key is a random number that you generate and keep secret. It is used for decrypting messages and generating digital signatures.

Your public key is derived from your private key and, as the name suggests, is designed to be publicly distributed. It's used for encrypting messages to you and generating digital signatures. Your address (where people sent transactions to) on the blockchain is typically derived from your public key.

If someone can guess or steal your private key, they have complete control of your account on the blockchain.

The security of public key cryptography is based on two things. The first is the secrecy of your private key. If someone can guess or steal your private key, they have complete control of your account on the blockchain. This allows them to perform transactions on your behalf and decrypt data meant for you. The most common way that blockchain is "hacked" is people failing to protect their private key.

The other main assumption of public key cryptography is that the algorithms used are secure. Public key cryptography is based off of mathematical "hard" problems, where performing an operation is much easier than reversing it. For example, it's relatively easy to multiply two numbers together but hard to factor the result. Similarly, it's easy to perform exponentiation but hard to calculate logarithms. As a result, it's possible to create schemes where computers are capable of performing the easy operation but not the hard one.

The security of these "hard" problems are why you'll often see articles about quantum computers breaking blockchain. Due to how quantum computers work, factoring and logarithms aren't much harder than multiplication and exponentiation, so traditional public key cryptography no longer works. However, other problems exist that are still "hard" for quantum computers, so the threat of quantum computers to blockchain can be fixed with a simple upgrade.

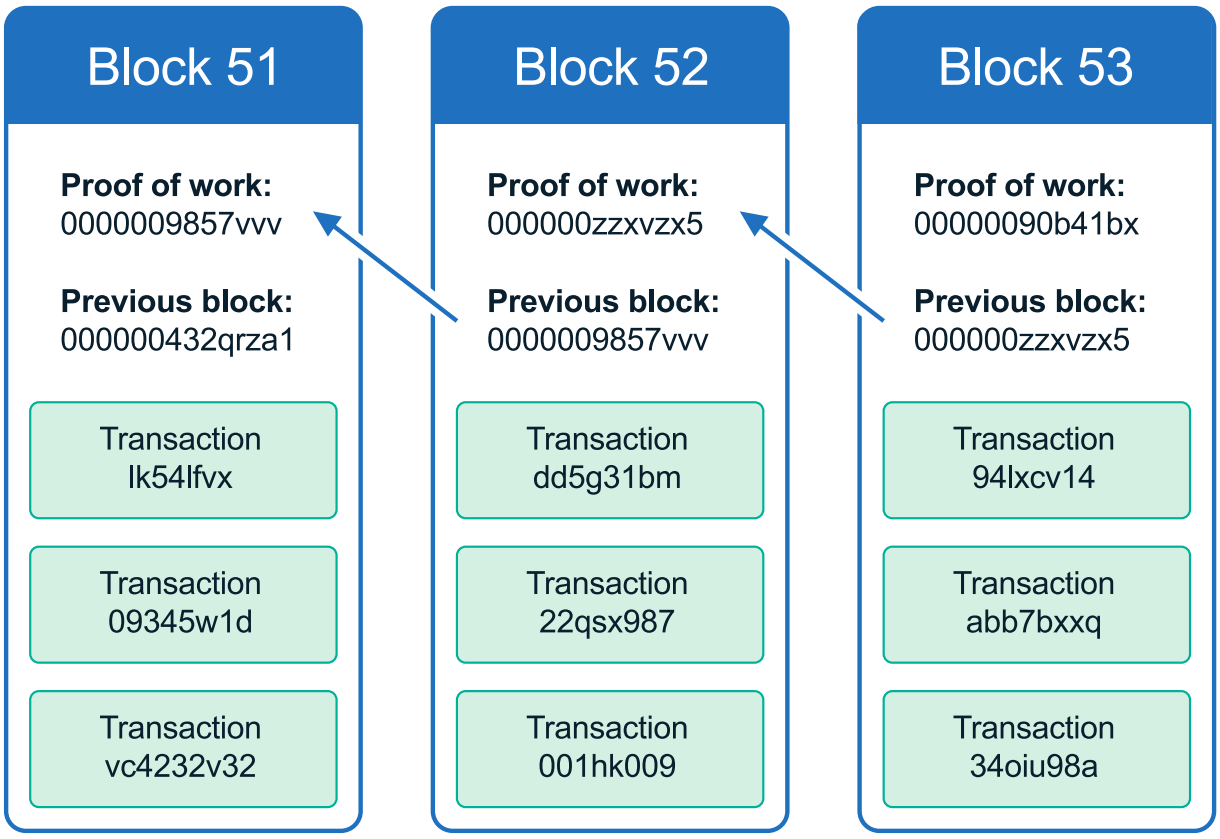
How Blockchains are Put Together

As its name suggests, the blockchain is a collection of blocks that are chained together to create a continuous whole. In this section, we explore how this works.

Blocks

The purpose of the blockchain is to act as a distributed ledger that stores data in a secure fashion. The blocks are the place where this data is stored.

The following image illustrates the basic structure of a block in a blockchain. We'll talk about every part of this image throughout the series, but for now focus on the green sections. Each green piece represents a transaction within the block. While a transaction may represent a literal transaction (i.e. a transfer of value) on blockchains like Bitcoin, this is not the only option. As we'll see later, smart contract platforms store other things (like computer code) as transactions as well.



The security of the blocks in the digital ledger depends on the security of public key cryptography. Every transaction and block in the blockchain is digitally signed by its creator. This allows anyone with access to the blockchain to easy validate that every transaction is authenticated (i.e. sent by someone who owns the associated account) and has not been modified since creation. The integrity and authenticity of the blocks in the chain is also assured by the digital signature of the block creator.

Chaining

Each block is equivalent to a single page in a bank's account ledger; it only represents a slice of the history of the network's history. In order to combine these slides into a continuous whole, the blockchain makes use of hash functions.

In the image above, you can see the hash functions linking each block together. Each block contains the hash of the previous block as part of its block header (the section not containing transaction data).

The fact that every block is dependent on the previous one is significant because of the collision-resistance of hash functions. If someone wanted to forge block 51 in the image, they have two options: find another version of block 51 that has the same hash or forge every block after 52 as well. The first is supposed to be impossible (due to collision resistance) and the other should be difficult or impossible since the blockchain is designed to make forging even a single block difficult (more on this later).

The security of the "chain" part of blockchain is based upon the collision resistance of the hash function that it uses. If someone can find a way to generate another version of block 51 that has the same hash, the immutability assumptions of blockchain break down and you can't trust that any transaction will remain in the distributed ledger.

Nodes

Nodes

Traditional network services work on a client-server model. To access the shared resource, you (the client) connect to a server and request the official version of the file. This makes synchronization easy (since the server knows the most recent version) but is very centralized. This can be problematic because it requires trust in the server and the server is vulnerable to Denial of Service (DoS) attacks.

Blockchain is designed to be a completely decentralized system. Every node in the blockchain network has the ability to keep a copy of the distributed ledger, and the the official version of the shared ledger is updated via blockchain consensus mechanisms (covered in detail in the fourth section).

What Do Nodes Do?

Nodes are a vital part of the blockchain ecosystem because they're the ones that do everything. As a decentralized peer-to-peer system, everyone acts as a combined client and server. As a result, the duties of nodes are protocol-specific (rather than software-specific) and numerous.

Protocol Not Software

Like many other Internet applications, a blockchain is a protocol rather than a specific piece of software. Instead of mandating that everyone run the same executable to use a service (like Skype), the only requirement is that nodes communicate based upon the rules of the service.

An example is HTTP, the protocol that defines how websites work. The structure and ordering of packets on the network is defined by the protocol, but no-one cares which software you're running. As a result, there are a couple of different web servers (Apache, IIS, etc.) and many different web browsers (Chrome, Firefox, Safari, etc.). These servers and browsers have agreed to follow the protocol, so they're able to communicate with one another with no issues.

Some blockchains are implemented using different software, while others have only one. When choosing blockchain software to run, it's always a good idea to cross-compare the options.

Common Node Tasks

The purpose of the node is to implement and operate the blockchain. Each node has the ability to store a complete copy of the distributed ledger, and, if they do, to update it based upon the consensus of the network as a whole. As a result, nodes can participate in a variety of activities including transaction processing, block creation, and ledger management.

Transaction Processing

One of the most common tasks that nodes have is transaction processing. Anyone connected to the blockchain network through the node will send their transactions to the node to be added to the distributed ledger. The node is responsible for sending these transactions on to the rest of the network as well as forwarding on any transactions that it receives from other nodes to its peers in the network.

Many attackers have found that it's more profitable to target the individual users.

Block Creation

The blockchain is updated by adding new blocks to the existing chain. These blocks contain the data stored on the distributed ledger, and someone needs to collect this information into the block and distribute it to the rest of the network. Since there is no centralized server in blockchain, this means that the nodes are responsible for this as well. Using a blockchain consensus algorithm, a node is selected as the next block creator. They perform the tasks of creating the next block and starting its distribution (and are rewarded for their trouble).

Ledger Management

Finally, nodes are responsible for ensuring that the distributed ledger is properly stored and accessible. Every node has the potential to store a complete copy of the distributed ledger. Since not all users of the blockchain network are nodes (i.e. some people just use Bitcoin for performing transactions or investments), these nodes may occasionally be asked to send a copy of certain parts of the blockchain to a user in order to verify that a transaction made it onto the distributed ledger.

Types of Blockchain Nodes

The role distinctions in the blockchain network aren't even as simple as node and not-node. In some cases, it's possible to have different types of nodes. For example, Hyperledger permits a huge amount of role specialization, allowing nodes to only do the portion of the work that they are most suited to.

One of the more common distinctions between nodes on the blockchain is full and lite nodes. As their name suggests, full nodes perform all of the job roles associated with being a node. These guys store a complete copy of the ledger and participate in consensus and block creation. A blockchain network needs a certain critical mass of full nodes in order to maintain its security and decentralization.

Lite nodes are designed to make it easy for someone to perform and verify transactions without doing everything that a full node does. In the previous section we talked about how the block headers are "chained" together using hash values. Since these headers summarize all of the transactions contained in a block, they are all you need for verification of blockchain integrity. Lite nodes download the headers and only request the actual transaction data if they want to verify that a certain transaction was included in the block. This reduces the storage and communications requirements of lite nodes at the cost of a bit of decentralization.

$1m Bitcoin being stolen by just one attacker in a matter of hours.

$20m Ether stolen via Remote Procedure Call (RPC).

Security of Blockchain Nodes

Nodes are the targets of most attacks on blockchain networks. While other attacks may have more name recognition (like 51% attacks), many attackers have found that it's more profitable to target the individual users. Some threats at the node level are security misconfigurations, phishing, and malware.

Security misconfiguration vulnerabilities occur when users modify the settings on their blockchain software without understanding the potential impacts. One example is a setting on a common Ethereum client that allowed external applications to communicate with wallet software via Remote Procedure Call (RPC). Attackers scanning for port 8545 were able to connect to the software and steal $20 million in Ether.

The decentralization of a blockchain network makes it more difficult to defend against certain network-level attacks

Phishing attacks are also extremely common for blockchain users. The Electrum wallet is especially known for being a target of phishing attacks, with over $1 million in Bitcoin being stolen by just one attacker in a matter of hours.

Finally, malware can be used on blockchain nodes for a variety of different purposes. Many of the attacks described in the remaining chapters can be performed using malware that targets the blockchain software on a node.

Securing Your Node

If you run a node on the blockchain, its security is completely under your control. Taking the appropriate steps to secure it like installing antivirus software, properly configuring it, and being aware of phishing scams can make a huge difference for your security and that of the blockchain network. The decentralization of a blockchain network makes it more difficult to defend against certain network-level attacks, but every secure node contributes to the health and security of the network.

Networks

Blockchain Networks

The blockchain is designed to store a trusted, shared distributed ledger. This ledger represents the history of the blockchain network, so the network level is an important one when discussing the blockchain ecosystem.

In the previous section we discussed the nodes and how they each maintain their own copy of the distributed ledger. Since the blockchain is designed to be trustless, no other node is going to implicitly trust any other node's copy. They need a way to agree on the state of the ledger (consensus), and, for that, they need a way to communicate: the network.

The Blockchain Peer-to-Peer Network

Blockchains use a different network architecture than most of the web services that we're used to. These services use a client-server architecture, where the server acts as a single source of ground truth, and the clients connect directly to it to upload or download application data. For example, when you use a webmail client like Gmail, your email doesn't go directly from your computer to the recipient's. Instead, you upload it to the Gmail servers and the recipient downloads it from a Gmail server to read.

This system is simple and effective, yet it relies on the Gmail server to be a trusted middleman in the process. Blockchain isn't big on trusted middlemen, so it uses a peer-to-peer network, where each node in the network communicates directly with other nodes. Most blockchain networks use a broadcast system where, if a node has five peers, every message that is received from one is sent to the other four. This way, messages percolate across the network over many paths, and no one has complete control over communications.

The main implication of the peer-to-peer model for blockchain networking is that the underlying network needs to be able to support it. Since every peer needs to be capable of connecting to every other peer, you can't effectively have a blockchain network distributed across a network with varying trust levels without compromising either blockchain or network security. Also, the "broadcast" communication style of the blockchain means that it requires a large amount of bandwidth to function properly. The inability to support this can have negative impacts on blockchain security and effectiveness.

Attacking the Blockchain Network

Many of the best-known attacks against blockchain systems are at the network level. Many people know that private key management is a problem and that smart contract vulnerabilities exist, but they'd be hard-pressed to even name the top ten most common smart contract vulnerabilities. On the other hand, Sybil attacks and 51% attacks are commonly mentioned in blockchain security-related posts.

In this section we'll discuss three network-level attacks on the blockchain: Denial of Service (DoS), Eclipse, and Sybil attacks. A 51% attack can also be considered a network-level attack, but we'll talk about it in the next section since it's most closely related to consensus.

Denial of Service Attacks

Blockchains are distributed, decentralized networks, so it seems like Denial of Service (DoS) attacks should be impossible. DoS attacks target a single point of failure (like a webserver) or a bottleneck in a system and attempt to overwhelm it in order to degrade the operations of the system. Since blockchain (theoretically) has no single points of failure, DoS attacks shouldn't be an issue. In practice, DoS attacks against the blockchain exist, but they attack temporary single points of failure or system bottlenecks.

One such bottleneck is the transaction capacity of the blockchain. Most blockchains create blocks with a fixed maximum size at a fixed rate. An attacker can create a large number of spam transactions and transmit them to the network (similar to a DoS attack on a webserver). If the network can't reliably identify them as spam transactions and ignore them, they'll be added to the blockchain, taking up space that could have been used by legitimate transactions. Worse, blockchains are "forever", so these spam transactions that make it onto the blockchain can take up storage space on nodes for the life of the blockchain.

An example of a temporary single point of failure is the creator of a given block. Different blockchains have different methods of choosing this person, but in the end, one node puts a block together, signs it, and transmits it to the network. In some schemes (like Proof of Stake), if a block creator misses their "slot" for creating a block, they forfeit it. If you can force someone to forfeit a block (i.e. by a traditional DoS attack), that block is never created and the network loses some of its potential capacity.

Eclipse/Routing Attacks

Eclipse and Routing attacks are two names for essentially the same attack. In an Eclipse attack, an attacker isolates a single node from the rest of the network by controlling all of its peer connections. In a Routing attack, the network is split up into two or more isolated groups. Both attacks can be used to facilitate a double-spend attack (by sending a different transaction to each isolated individual/group) or a 51% attack (by filtering the victim's view of the network state so that they mine a version of history that's in the attacker's favor).

Eclipse and routing attacks can be performed by a variety of different means. External to the blockchain ecosystem, an attacker can control a node's connection to the network using malware or any other traditional means of performing a Man-in-the-Middle (MitM) attack. A study found that Bitcoin is especially vulnerable to BGP routing attacks, where the attacker convinces computers that the best route from A to B is through them.

Internal to the blockchain ecosystem, an attacker can perform these attacks by controlling all of a node's connections to their peers. Since blockchain networks are not fully connected (nodes only connect to a small number of other nodes), it's possible that a node can only be connected to peers controlled by an attacker.

It doesn't matter how the attacker controls the node's connection to the blockchain network as long as control is absolute. If this is true, the attacker may be able to selectively drop packets from other users or send mutually exclusive versions of transactions from their own addresses to drive the isolated groups' versions of history apart. When the attack is completed, the longest block rule means that whichever version of the blockchain is shorter will be discarded (which is perfect for a double-spend attack).

Sybil Attacks

A Sybil attack is a simple network-level attack used to facilitate other attacks. In a Sybil attack, the attacker creates and maintains a large number of accounts on the blockchain network. This can be useful when performing an Eclipse/Routing attack since, if the attacker controls most of the nodes accepting connections when a node is looking for one, there is a high probability that the node will only choose attacker-controlled connections.

Consensus

Blockchain
Consensus

Blockchains are designed to be distributed, decentralized networks. Part of this includes removing the central authority used in many other systems. In a traditional financial system, banks centralize power by maintaining control of the ledger that states how much value is stored in each account. If a dispute arises over the ledger, the bank has the final authority to decide what the authoritative version is.

Blockchain is designed to remove centralizing authorities like banks. Instead, the blockchain network maintains a shared, decentalized ledger with each node in the network maintaining a copy and updating it as each new block is created.

The challenge with this is ensuring that all nodes make the same updates to their copies of the ledger with each block. Since the network does not have a consistent authority to create the official version of the ledger, it chooses a temporary authority to create and share each block. The mechanism for accomplishing this is called the blockchain consensus algorithm.

Fundamentals of Consensus

The job of the consensus algorithm is to ensure that control of the blockchain is decentralized so that no one user has the ability to control the network. The means by which this is accomplished is through making control of the blockchain network dependent on control of a scarce resource.

No matter what consensus algorithm you choose, it boils down to the fact that control of a scarce resource equals power on the blockchain. In Proof of Work, this resource is computational power. In Proof of Stake, it's the blockchain's cryptocurrency.

The logic behind using a scarce resource as an analog to power on the blcokchain is that it enables the use of economic incentives to protect the blockchain. The Law of Supply and Demand says that, if there is increased demand for a resource with a limited supply, then the price increases.

When an attacker tries to gain control of a blockchain network (to perform a 51% attack or similar), they need to acquire more of the scarce resource to do so. As a result, they increase the demand for the resource, which increases the price to acquire it. Hopefully, the cost to acquire enough of the resource to perform a successful attack will be beyond the attacker's resources. If not, we have successful 51% attacks against blockchains, which has certainly happened on smaller cryptocurrency networks.

How Common Algorithms Implement Consensus

When Satoshi Nakamoto created Bitcoin, it was the only blockchain in existence. The Bitcoin whitepaper described the Proof of Work consensus algorithm used on the Bitcoin network. Since then, many other consensus algorithms have been developed for different blockchain implementations. Of these, Proof of Stake also receive a lot of attention, partly due to its presence on the Ethereum roadmap.

Proof of Work

Proof of Work is the original consensus algorithm, and, as its name suggests, it involves making people do work. In Proof of Work, miners are the ones attempting to create a new block. The way that the block creator is selected is by implementing a race where the winner creates the block (and earns the associated rewards).

This race involves creating a valid block, where the condition for validity is that the header of the block hashes to a value less than a given threshold. Due to the properties of hash functions, the best way of accomplishing this is by random guessing. As a result, the miners in the network try random hashes until one stumbles across a nonce that creates the desired hash output. The first miner to find a valid block then transmits it to the rest of the network to build the next block on top of.

The main issue with Proof of Work is that the criteria for block creation is the ability to create a valid block. There is nothing to say that two different miners can't find different versions of the block around the same time. If this occurs, a divergent blockchain may be created with different parts of the network building on top of different blocks. Blockchain resolves this using the longest block rule, which says that, in a conflict between two versions of the blockchain, the longer one should be accepted.

Proof of Work also tries to minimize the probability of divergent blockchains using the concept of difficulty. The threshold value that a valid block header's hash must be less than can be updated in a distributed fashion. The difficulty is updated at regular intervals so that the creation of blocks (with the current computational power of the blockchain network) occurs at the desired block rate.

Proof of Stake

Proof of Stake takes a different approach to securing the blockchain using a scarce resource. Instead of using scarce computational power (like Proof of Work), Proof of Stake uses the blockchain's scarce cryptocurrency.

Proof of Stake works a lot like investing in a company. By giving some of your money to a company, you have the right to receive investor dividends. In Proof of Stake, you promise not to spend a portion of your cryptocurrency (or stake it) in exchange for the chance to be a block creator (and earn the associated rewards).

The mechanics of how block creators are selected based on stakes varies based upon the implementation. In some implementations, the probability of being selected is directly proportional to the size of the user's stake. In others, the concept of coin age is introduced, where stakers who have not been selected to create a block in some time have an increased probability of being selected. Regardless, control of more staked cryptocurrency in Proof of Stake equates to increased control over the blockchain.

One issue with Proof of Stake is the potential for a user to create multiple versions of the same block. Since the only criteria for a block to be valid is a signature by the chosen block creator, it's possible for a user to sign multiple versions of the same block. In fact, this is one place where blockchain incentives break down since, if presented with two versions of the blockchain to build upon, it's in the block creator's best interest to build on both to ensure that whichever version eventually wins out includes the block that pays them their block reward.

Attacking Consensus

Consensus mechanisms are the key to controlling the blockchain. As a result, many attacks on the blockchain are based upon gaining this control. If successful, an attacker can perform a double-spend attack, which allows them to complete one transaction and then remove it from the ledger at a later date. Some attacks against consensus have been known from the beginning (like the 51% attack), while others (like long-range attacks) were developed later.

51% Attacks

51% attacks are probably the simplest way to attack a Proof of Work blockchain and occur when the economic incentives of the blockchain don't work. Under the longest block rule, every benign node is obligated to choose the longer option when presented with two contradictory versions of the blockchain. If an attacker has the ability to create the longer version at will, then they control the blockchain.

In Proof of Work, this is accomplished by controlling over half of the computational power of the blockchain network. Since creation of valid blocks requires randomly searching the space of potential options, whomever can search the space more quickly can create blocks faster. Similar attacks are possible on Proof of Stake, but it requires a greater level of control over the scarce resource. In Proof of Work, you need 50% of the computational power to have a 100% chance of finding the next block. In Proof of Stake, you need 100% of the staked cryptocurrency to have a 100% chance of forging the next block. Since this is unlikely, an attacker trying to control a Proof of Stake blockchain needs to accept the possibility of failure.

Long-Range Attacks

Long-range attacks can be used on Proof of Stake blockchains to give an attacker the controlling portion of the staked cryptocurrency necessary to attack the consensus algorithm. In this attack, the attacker creates a divergent version of the blockchain all the way back to the genesis block (this assumes that they have a stake in the genesis block).

On their divergent blockchain, the attacker creates a new block whenever they are selected as the block creator. Since they are the only ones creating blocks, they're the only ones receiving block rewards. Over time, the attacker has the controlling stake in the divergent blockchain.However, the divergent blockchain will only be accepted if it is longer than the "true" version of the blockchain. Since the attacker can only create blocks on their version when it's their turn, their divergent blockchain will fall behind the main chain whenever a benign user is selected to create a block. While this happens less frequently as they control more of the stake, the attacker's chain is significantly behind in the beginning.

In order to catch up, the attacker deliberately passes up their opportunities to create blocks on the main chain. Between these missed blocks and ones missed by natural causes (or due to a Denial of Service attack on the chosen block creator), the attacker's chain has the opportunity to slowly catch up to the main chain. When this occurs, the attacker can publish their malicious divergent blockchain and gain control of the blockchain.

Smart Contracts

Blockchain
Smart Contracts

In the beginning, blockchain was designed to replace the financial system. The distributed, decentralized ledger maintained by the blockchain network is used to record the transactions performed using the blockchain financial system. As a result, cryptocurrencies like Bitcoin can implement complete, trustworthy financial systems without a central authority (like a bank).The distributed, decentralized ledger offered by blockchain technology is useful for more than just recording financial transactions. Smart contract platforms are designed to run a Turing-complete computer on top of the blockchain, allowing smart contracts to fulfill a variety of different functions.

Introduction to Smart Contracts

Smart contract platforms use the underlying blockchain technology but modifies it to use it to run arbitrary, third-party programs on top of it. Instead of transactions including actual financial transactions, they include computer instructions designed to be run by the blockchain's virtual machine.

Since the blockchain network is distributed and decentralized, there is no central computing platform that runs the code and updates the state of the smart contract platform with the result. Instead, each node in the network runs its own copy of the virtual machine and executes the code contained in the transactions in each block of the blockchain. Since code is designed to be deterministic and is organized into a block before execution, the network is able to remain synchronized at all times.

Smart Contract Security

The blockchain landscape is fragmented, and so is the landscape of smart contract blockchains. The basic blockchain solution has been adopted and adapted in many different ways, making it difficult to create a definitive list of smart contract vulnerabilities.

Since Ethereum is the best-established smart contract platform, many lists of smart contract vulnerabilities focus on this. The Decentralized Application Security Project has compiled a list of the most common smart contract vulnerabilities on the Ethereum platform, which are explored here.

Reentrancy

Reentrancy is probably the most famous of the Ethereum smart contract vulnerabilities. Exploitation of this vulnerability in The DAO smart contract caused Ethereum to break its blockchain's immutability

and rewrite history to erase the attack. This controversial decision caused a split in the Ethereum network that created the Ethereum Classic cryptocurrency.

Reentrancy is possible in Ethereum due to the existence of payable fallback functions. These functions are designed to run when value is sent to the smart contract, allowing it to update its internal ledger, perform some functions, etc.

Vulnerabilities like integer overflows and underflows are nothing new with blockchain... some smart contract programming languages are now vulnerable.

The issue with this setup is when a vulnerable smart contract function that the attacker can control (like a refund function) can be forced to call a malicious fallback function. Vulnerable functions use the following control flow:

Check transaction validity → Perform send → Update internal ledger

With this control flow, the malicious fallback function is run as a part of step 2, before the vulnerable function updates state. If it calls the vulnerable function again, the transaction will still be considered valid (since the state isn't updated) and run again, allowing the attacker to withdraw twice as much value as approved.

Access Control

Some smart contracts are designed to have protected functionality. For example, you can implement wallets as smart contracts where anyone can send value to it but only the owner can extract value from it.

Some of these smart contracts have a function for claiming ownership, where the owner of the smart contract (and the one permitted to call protected functionality) is set to the person who called the function. The issue with these functions is that sometimes people forget to set the function to check that this is the first time the function is called. If they fail to do so, the owner is whoever called the function last, not first.

Arithmetic

Arithmetic vulnerabilities like integer overflows and underflows are nothing new with blockchain. They've existed in software programming for some time and have only become less common due to the existence of programming languages that make them impossible (like Python). Unfortunately, some smart contract programming languages are now vulnerable.Arithmetic vulnerabilities occur when certain variable types are misused. Integer overflow vulnerabilities occur when a programmer uses too small of a variable to store a value. Underflow vulnerabilities occur during switches between signed variables (where a one in the most significant bit means negative) and unsigned variables (where a one in the most significant bit means a large, positive number). Performing subtraction with unsigned values always results in a positive number, which can be problematic since these tests are often performed to check the validity of a transaction.

Unchecked Return Values

An Ethereum-specific feature that can trip up novice smart contract developers is the fact that it does not have a consistent means of indicating when low-level functions fail. Some low-level functions throw an error if they fail, which terminates execution. Other ones return a value of false and allow the code to continue running. If an programmer assumes the first case for a certain function and doesn't check function return codes, it's possible to put their code in an unexpected (and potentially invalid) state.

Denial of Service

Just like the underlying blockchain can be vulnerable to Denial of Service attacks, smart contracts can also be rendered non-operational by a malicious (or benign) user. Denial of Service attacks on smart contracts can be accomplished in a variety of different ways. One way to attack a smart contract is to exploit an access control vulnerability. Well-designed smart contracts include a self-destruct function that would render the contract unusable if an attacker gains access to and executes this function.

Bad Randomness

Smart contracts often need access to random numbers. In fact, many smart contracts are designed to implement gambling games, so they need the ability to generate secret random numbers.

There are several means of generating random numbers in code, and many of these are considered "best practice" in traditional applications. However, the blockchain environment is different, making the following means of generating randomness insecure:

'Secret' Values: Like seeding a pseudo-random number generator, some smart contracts use 'secret' values to create randomness. However, everything is public on the blockchain, so an attacker can observe this value and predict the 'random' values.

'Secret Code': Using a proprietary algorithm for generating random numbers is not a great idea but it often works. However, it fails on blockchain for the same reason as the 'secret' values.

External Input: Using external sources of entropy is a common method of generating randomness in traditional applications. However, any source of entropy visible to one smart contract is visible to any other, making it easy to observe and exploit.

In the end, the best way to generate random numbers on the blockchain is to use an external source of randomness that the smart contract can query. However, this has to be done carefully to ensure that malicious smart contracts can't see it as well.

Race Conditions

In traditional programming, race conditions are when two or more threads are competing for resources, and the behavior of the program is dependent on which one gets there first. In the blockchain, multiple transactions may be competing for recognition by a smart contract and the result depends on whichever transaction is processed first.

For example, a contest may exist where the first person to solve a puzzle wins some prize. A benign user solves the puzzle and submits their solution to the smart contract as a transaction. However, transactions are not instantly processed, are publicly visible before processing, and are organized for processing based upon transaction fees. If an attacker sees the user's solution, copies it, and submits a transaction with a higher transaction fee, the attacker is likely to win the contest without doing any work to solve the puzzle.

Timestamp Dependence

Another way that these contests can go wrong is if they depend on the current time on the blockchain as a condition for the contest. For example, a smart contract may run a contest where the first submission after midnight on a certain day is the winner.

On the blockchain, the current time is the time of the most recent block, and this is set by the block creator. Further, there is some wiggle room (often up to two hours) in timestamps to deal with propagation delay, non-synchronized clocks, etc. (in fact block timestamps don't even have to be in order). An attacker who manages to create a valid block with a timestamp of before midnight but within the acceptable window can win this contest before anyone else tries to play.

Short Addresses

Short address vulnerabilities in Ethereum are caused by variable sizes, how arguments to a function are stored in memory, and how Ethereum pads arguments that are too short. In this attack, the attacker calls a vulnerable smart contract function designed to send value to a certain address (like the refund function from the reentrancy vulnerability). In this call, the attacker deliberately sends a destination address that is one byte too short and a value of the correct size. The function checks the value and, if the transaction is valid, calls a function to transfer the value.

This transfer function specifies the size of its arguments and expects a destination address of a given size. As a result, it fills the address variable with the provided address and the first byte of the provided value. Now, the value is too small, so Ethereum zero-pads it on the right, effectively multiplying it by 256. As a result, if the new destination address is controlled by the attacker (which they can assure before they perform the attack), they receive 256 times more value than the vulnerable function authorized.

Unknown Unknowns

The final smart contract vulnerability included in the DASP list was unknown unknowns. Blockchains in general, and smart contract platforms in particular, are a relatively new technologies. It is extremely likely that new vulnerabilities will be discovered and take the top slots for smart contract vulnerabilities in future years.

Extensions

Blockchain Extensions

Blockchain technology provides users with a number of advantages not present in traditional systems. Blockchains are the first fully distributed and decentralized system that is capable of maintaining a shared, trusted ledger. This allows a network to keep a record of its history and be confident that a malicious user or users is not capable of modifying this history to their own benefit.

However, blockchain technology isn't perfect. Bitcoin was originally designed to replace traditional payment systems (like credit cards); however, by itself doesn't have the ability to do so. Blockchain technology has limitations, and blockchain extensions have been developed to help mitigate or eliminate these.

Limitations of Blockchain

Blockchains has a very specific structure. Due to the need for the network to remain synchronized and for the network to validate all transactions, transactions cannot be continuously added to the distributed ledger. Instead, transactions are organized into blocks, which are added to the distributed ledger at regular intervals. This design limits the speed and capacity of the blockchain solution.

The speed at which transactions are added to the distributed ledger is severely limited on the blockchain. Blockchains typically have a target block rate, that is enforced at some level by their consensus algorithm. For example, Bitcoin has a block rate of 10 minutes, meaning, with the three block rule, you may have to wait half an hour before a transaction is considered trustworthy. This compares unfavorably with credit cards, where a "slow" transaction is done in a minute.

Blockchains also have an issue with a maximum capacity. In addition to the set block size, many blockchains have a set maximum block size designed to protect against Denial of Service (DoS) attacks. With fixed-size blocks created at fixed intervals, the blockchain can only process so many transactions in a time period, and this capacity is often far below that of the payment card system.

Blockchain Extensions

Some distributed ledger technologies have abandoned the blockchain data structure in order to address these problems. For example, IOTA uses a directed acyclic graph (DAG) as its underlying data structure, which greatly increases its transaction speed and capacity. Some blockchains make small protocol tweaks (like increasing the block rate) to improve transaction speed and capacity. And some blockchains have begun leveraging blockchain extensions to help address these issues while maintaining the original design of the blockchain.

Sidechains

Sidechains are primarily designed to increase the capacity of the blockchain by offloading some transactions to a standalone blockchain. There are a few different implementations of sidechains, but a common one is to "peg" a sidechain to a parent blockchain. With pegged blockchains, a user on one blockchain can send tokens to an "output address", and the equivalent amount of tokens will be released onto the sidechain. Pegs are bidirectional, so the user can return to the original blockchain at will.

One benefit of sidechains is the increase in capacity for the original blockchain. Since transactions performed on the sidechain are not recorded in the blocks of the main blockchain, the total capacity of the system is increased.

Sidechains can also be used to address specific deficiencies of the parent blockchain. For example, a sidechain could have a faster block rate than the parent chain, increasing the system's transaction speed. Alternatively, sidechains can increase the capabilities of the system, like the Rootstock sidechain that plans to add smart contract functionality to Bitcoin.The main security consideration of sidechains is that the sidechain is a completely distinct system from the main chain. It needs to have its own means of securing consensus, through a large pool of miners, stakers, etc. Otherwise, a hack of the sidechain could affect the quality of its peg with the main chain and its users' ability to switch back and forth.

State Channels

Another blockchain extension that has been getting a lot of press is the state channel. The most famous state channel system is probably the Lightning Network running on the Bitcoin network. However, other state channel implementations run on other blockchains under different names. State channels function as a second-level protocol that runs on top of a traditional blockchain implementation. A state channel is a direct connection between users of the blockchain network. They establish the channel using a traditional blockchain transaction that establishes the balance that each has contributed to the channel (i.e. 1 BTC apiece). After the channel is established, payments are made by creating mutually signed assertions regarding the balance of value in the channel (i.e. .75 BTC and 1.25 BTC). The channel can be closed down at any time, and another blockchain transaction is created using the most recent balance assertion to place the correct amount of cryptocurrency in each participant's blockchain account. The main advantages of state channels are transaction speed, scalability, and privacy. Transactions only require the channel participants and can be completed near-instantaneously. However, if a channel becomes too unbalanced, it may be impossible to make a payment. This is where the network of state channels can be very useful since transactions can pass through different paths to rebalance channels or perform transfers between unconnected parties. The main security consideration of state channels is that transactions are backed by the blockchain but not recorded on it. State channel transactions are private to the recipients, and the blockchain network has to trust that all transactions made over them are legitimate. However, the point-to-point nature of state channels protects against double-spend attacks since the value stored in one channel is unique to that channel and cannot be used to open and perform transactions in other channels.

The Distributed Ledger Universe

This eBook was designed to be an introduction to blockchain technology with a focus on blockchain security. However, the solutions discussed in these articles are not the only ones out there. Other distributed ledger implementations (like DAGs and hashgraphs) use different data structures and have different security properties. Also, the blockchain can be extended using external devices that interact via APIs or smart contracts. When designing a distributed ledger solution, it's important to consider all of the available technology and the security considerations associated with it.

GhostVolt is the smart encryption tool built for teamwork.

Manage your team's workflow and all your digital assets in an secure environment with full asset restriction, control, auditing and reporting.

Visit **GhostVolt.com**