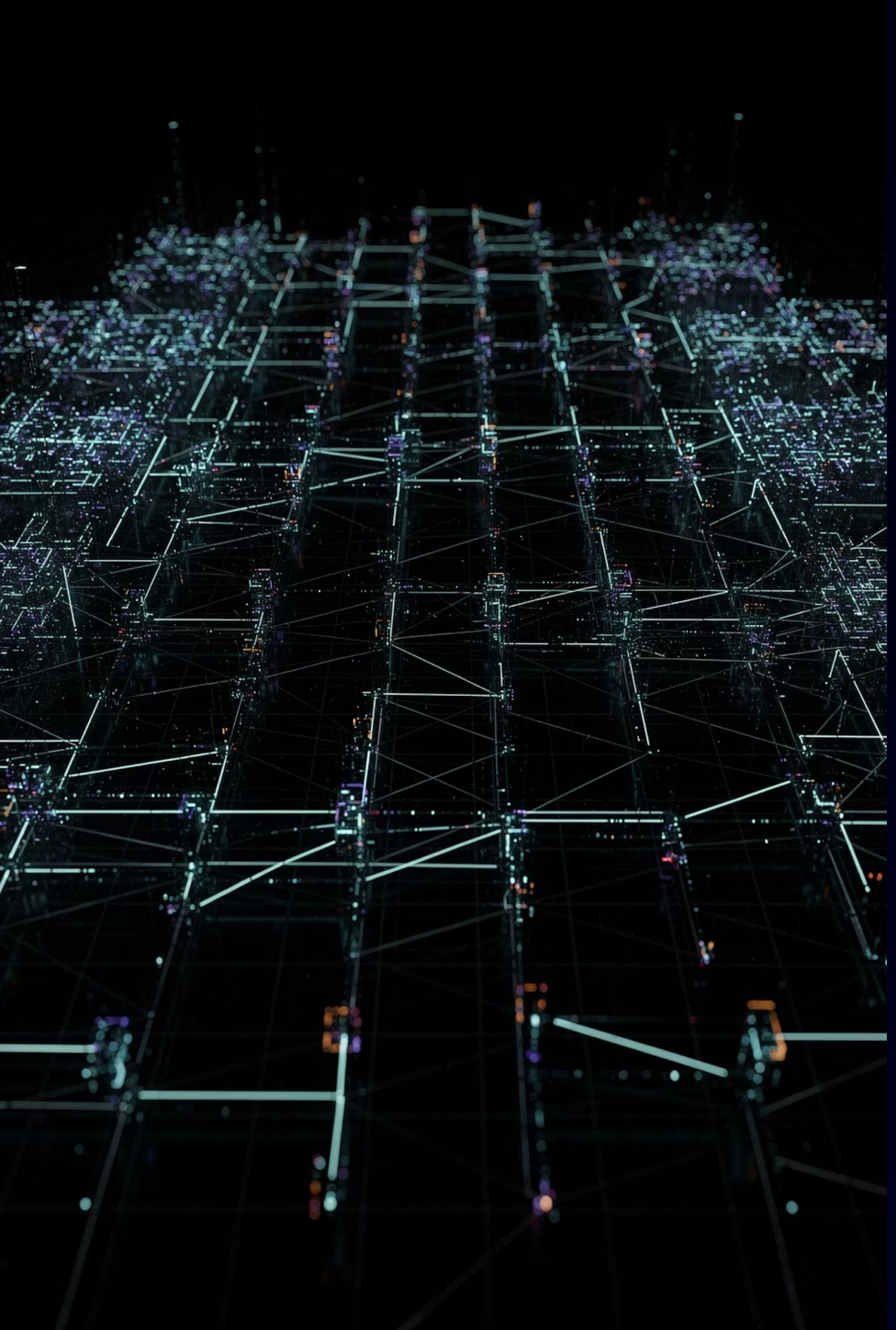


Blockchain: The Next Attack Surface

A Security-First Introduction for Cyber Defenders

Michael "0xObsidianEnoch" Lathan Jr. | March NEbraskaCERT Cyber Security Forum (CSF)

Why This Talk Exists

Moving Into Infrastructure

Blockchain is no longer theoretical: it is being embedded into financial rails, supply chains, identity systems, and government platforms.

Attackers Already Understand It

Adversaries are exploiting wallets, bridges, exchanges, and smart contracts at scale. The tooling and techniques are mature.

Many Defenders Do Not

Most cybersecurity teams still lack a working mental model of how blockchain actually functions as a security system.

That Gap Creates Risk

When defenders cannot reason about an attack surface, they cannot protect it. This talk closes that gap.

What Cyber Teams Usually Get Wrong

These misconceptions are common, and they are dangerous. Each one represents a blind spot that attackers actively exploit.

"Blockchain is just crypto"

Blockchain is a distributed systems and cryptographic engineering problem. The speculative asset layer is a tiny slice of a much deeper technical stack.

"Blockchain is secure by default"

The base protocol may be robust, but the ecosystem around it: wallets, apps, APIs, and bridges, is full of conventional and novel vulnerabilities.

"The cryptography handles everything"

Cryptography secures specific operations. It does not protect against misconfiguration, social engineering, or application-layer flaws.

"This is not relevant to defenders"

If it touches your organization's assets, supply chain, or infrastructure, it is absolutely relevant to your threat model.

Core Thesis

Blockchain security is not a single property; it is a product of multiple interdependent layers. A weakness at any layer can compromise the entire system.

Applications & Users

Wallets, dApps, exchanges, and human behavior

Infrastructure

Nodes, networks, and operating environments

Protocols

Consensus rules, transaction validation, peer communication

Data Structures

Blocks, headers, Merkle trees, transaction format

Cryptography

Hash functions, digital signatures, public key primitives

What You Will Leave With

Explain Blockchain in Security Terms

Describe how blockchain functions as a layered trust system without resorting to hype or financial framing.

Identify Major Attack Surfaces

Map the real attack paths across cryptography, infrastructure, consensus, contracts, and the application layer.

Apply STRIDE to Blockchain

Use a familiar threat modeling framework against an unfamiliar technology stack with layer-aware precision.

Understand Where Real Losses Happen

Move beyond protocol theory to the operational reality: most losses occur around the chain, not in it.

The Trust Shift

Blockchain does not eliminate trust; it relocates it. Instead of trusting a bank, a clearinghouse, or a government ledger, participants trust the protocol rules and distributed validation process.

Key insight for defenders: Trust still exists: it has simply been encoded into software and mathematics. That software can have bugs, and those mathematics can be misimplemented.

Goals of Blockchain Technology

Pseudonymous Identity

Participants interact via public key addresses, not legal names. Identity is discoverable but not immediately obvious.

Decentralization and Fault Tolerance

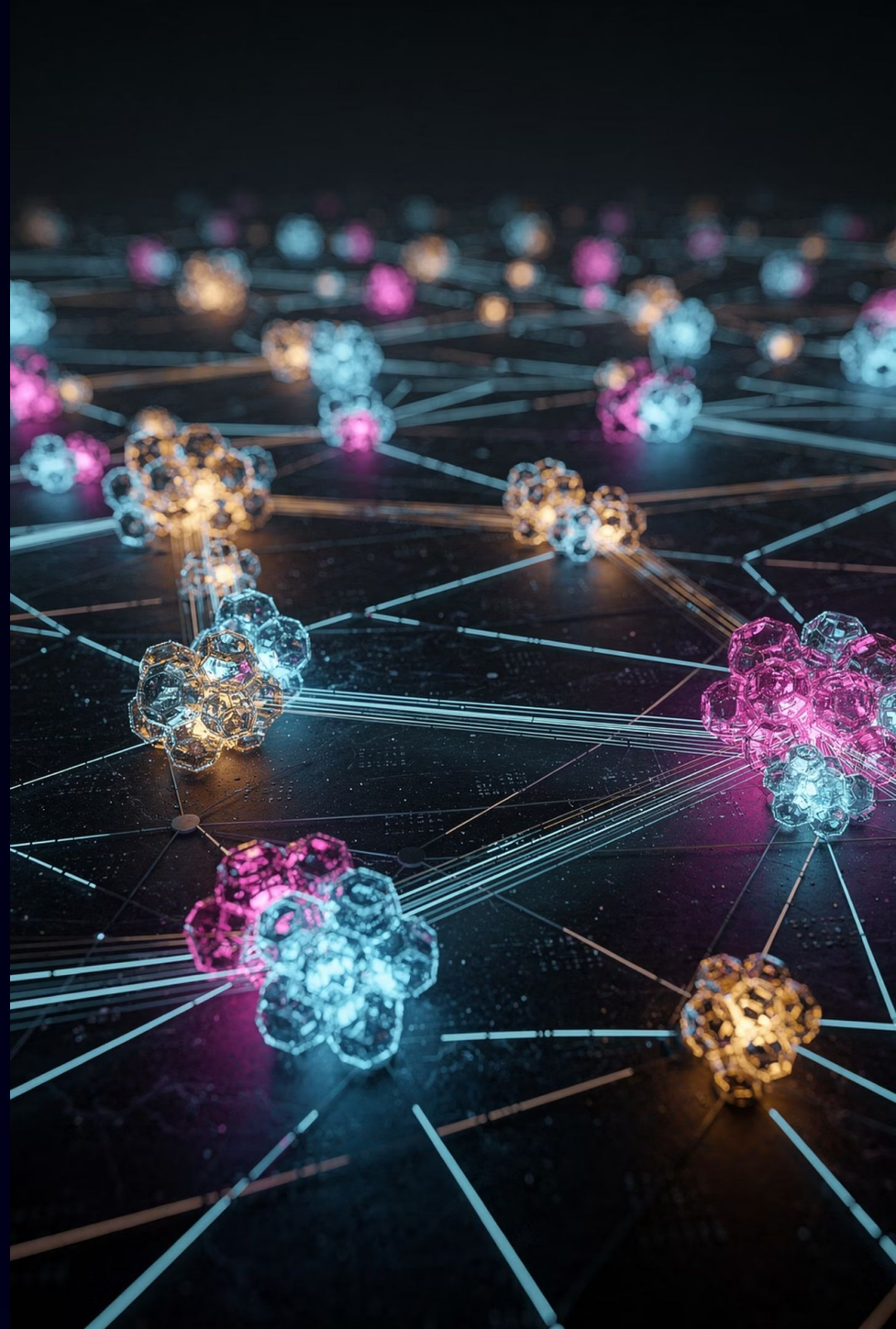
There is no single point of failure or control. The network continues operating even when individual nodes go offline or behave maliciously.

Immutability and Transparency

Once confirmed, records are computationally expensive to alter. The ledger is open for inspection by any participant.

Trust Minimization

Participants can verify state transitions independently. No single intermediary must be trusted to report honestly.



Goal Tensions

Blockchain design is a constant exercise in security trade-offs. Every goal pulls against another. Understanding these tensions is essential for realistic threat modeling.

Transparency vs. Privacy

Open ledgers enable auditability, but they expose transaction graphs to surveillance and analysis.

Decentralization vs. Performance

More validators mean slower consensus: high throughput typically requires concentrated control.

Immutability vs. Flexibility

Bugs baked into immutable contracts cannot be easily patched; however, upgradeability reintroduces trust assumptions.

Openness vs. Compliance

Permissionless networks resist censorship, but they conflict with AML, KYC, and regulatory requirements.

Pseudonymity, Not True Anonymity

This distinction matters for defenders and investigators alike. Blockchain addresses are permanent public identifiers, and the ledger is a permanent public record.

What Pseudonymity Provides

- A public address not directly tied to a legal name
- Ability to generate new addresses per transaction
- Separation between on-chain and off-chain identity

What It Does Not Provide

- Ledger analysis can cluster addresses and trace flows
- Metadata (IP address, timing, exchange KYC) leaks identity
- Public visibility enables surveillance as easily as auditing
- Once linked to a real identity, the entire history is exposed

Chain analysis firms routinely de-anonymize high-value actors. The ledger is forever: any future link exposes all past activity.

Decentralization Is a Spectrum

Decentralization is a design property that can be undermined in practice. Defenders should evaluate actual distribution, not theoretical architecture.

Validator Concentration

A small number of validators controlling majority stake can coordinate to censor transactions or rewrite history.

Mining Pool Dominance

In PoW networks, a handful of large pools routinely control more than 50% of hash rate, creating latent 51% attack risk.

Hosting Concentration

Most nodes run on a few cloud providers. A targeted outage at AWS or Azure can fragment entire networks.

Governance Capture

Protocol upgrade processes can be influenced by wealthy token holders or well-organized minorities, centralizing control of the rules themselves.

The Blockchain Stack from the Bottom Up

Thinking in layers is how security professionals reason about complex systems. Blockchain is no different: each layer has its own trust assumptions, attack surfaces, and defensive controls.

Public Key Cryptography as Blockchain Identity

In blockchain, identity is not a username or certificate; it is a cryptographic key pair. This is the foundational identity model the entire system is built upon.

Signature = Proof of Authorization: A valid signature proves the private key holder authorized a transaction. It does not prove identity in the legal sense, only cryptographic control.

Private Keys Are the New Root Passwords

Compromise a private key, and you have complete, irrevocable authority over everything that key controls. There is no password reset; there is no account recovery hotline.

Transfer Assets

Move all funds to attacker-controlled addresses instantly and irreversibly.

Impersonate the User

Sign any transaction as the legitimate owner with no technical distinction from authorized use.

Exercise Governance Rights

Vote on protocol upgrades, treasury allocations, and parameter changes on behalf of the victim.

Call Privileged Functions

If the key has admin rights on a smart contract, an attacker can drain funds, pause protocols, or change ownership.

What Digital Signatures Actually Do

Digital signatures are a precise cryptographic tool. Understanding their exact function prevents security misconceptions and helps identify where additional controls are needed.

What Signatures Provide

Authorization: Proves the key holder approved this specific transaction

Integrity: Any modification to the signed data invalidates the signature

Authenticity: Links the action to a specific key, supporting non-repudiation

What Signatures Do Not Provide

Signatures do **not** encrypt transaction contents; data is visible on-chain.

Signatures do **not** prove human intent; they only prove key possession.

Signatures do **not** protect against key theft before signing.

- Compromise the key, and the attacker's signatures are indistinguishable from legitimate ones.

Hard Math Behind Public Key Crypto

The security of public key cryptography rests on mathematical problems that are believed to be computationally infeasible to solve in reverse. "**Believed**" is the operative word: these are assumptions, not proofs.

1

Integer Factorization

Multiplying two large primes is easy. Factoring the result back into those primes is computationally prohibitive; this is the basis of RSA.

2

Discrete Logarithms

Computing g^x mod p is easy. Recovering x from the result is hard; this is the basis of classic Diffie-Hellman and DSA.

3

Elliptic Curve Discrete Logarithms

The ECDLP variant on elliptic curves is used in secp256k1, which is the curve behind Bitcoin and Ethereum key pairs.

Security depends entirely on these assumptions holding. Advances in quantum computing, such as Shor's algorithm, threaten all three. Post-quantum migration is an active area of concern.

Why ECC Matters

Elliptic Curve Cryptography (ECC) is the dominant choice for blockchain key pairs. Its efficiency advantages over RSA at equivalent security levels make it practical for high-volume transaction signing.

256

Bit Key Size

ECC offers security comparable to 3072-bit RSA with a 256-bit key. Smaller keys mean faster operations and less data on-chain.

~10x

Faster Signatures

ECC signature generation and verification are significantly faster than RSA at equivalent security levels; this is critical for high-throughput networks.

128

Bit Security Level

secp256k1 provides approximately 128-bit security: it is sufficient today, but quantum computers running Shor's algorithm would break this.

Weak Key Generation = Weak Security

The strength of the entire cryptographic model collapses if keys are generated poorly. This is one of the most reliably exploited attack paths in the blockchain ecosystem.

Bad Entropy Sources

Keys generated with insufficient randomness are predictable. Embedded systems, virtualized environments, and poorly seeded PRNGs are common failure points.

Predictable RNG

Weak or deterministic random number generators have led to real-world key recovery attacks. The 2013 Android Bitcoin wallet vulnerability is a documented case.

Unsafe Wallet Tools

Third-party key generation libraries with subtle bugs can silently produce weak keys that look valid but fall within a predictable keyspace.

Broken Key Generation Path

Any compromise in the pipeline, from seed phrase generation to key derivation, can expose the final key regardless of the algorithm's theoretical strength.

Hash Functions and Ledger Integrity

Cryptographic hash functions are the connective tissue of blockchain data structures. They are used pervasively across every layer of the stack.

Block Linking

Each block header contains the hash of the previous block header, creating a tamper-evident chain. Altering any historical block breaks all subsequent hashes.

Merkle Trees

Transactions in a block are hashed pairwise up to a single Merkle root. This enables efficient inclusion proofs without downloading the full block.

Signature Workflows

Before signing, transaction data is hashed. The signature is computed over the hash, not the raw data; this binds the signature to a compact, fixed-size commitment.

Security Assumptions of Hash Functions

Hash functions used in blockchain (SHA-256, Keccak-256) must satisfy three core security properties. If any property fails, specific attack classes become feasible.

1

Preimage Resistance

Given a hash output H, it must be computationally infeasible to find any input M such that hash(M) = H. This protects against reversing hashes to recover data.

2

Second-Preimage Resistance

Given input M1, it must be computationally infeasible to find a different M2 such that hash(M1) = hash(M2). This prevents targeted substitution attacks.

3

Collision Resistance

It must be computationally infeasible to find any two distinct inputs M1 and M2 with the same hash output. This protects Merkle tree integrity and signing workflows.

Practical infeasibility matters: MD5 and SHA-1 are broken for collision resistance. SHA-256 and Keccak-256 remain secure today, but algorithm choices in implementations vary.

Merkle Trees and Inclusion Proofs

Merkle trees are the data structure that makes blockchain verification scalable. They allow a lightweight client to verify that a transaction is included in a block without downloading the entire block.

To prove Tx1 is included, a verifier only needs Hash(Tx2) and Hash(H3+H4), rather than the full transaction set. This enables efficient, trustless inclusion verification at scale.

Transactions as Signed Data Structures

A blockchain transaction is a precisely structured, cryptographically signed message that authorizes a specific state change. Each field serves a security function.

Inputs

References to prior unspent outputs or account balances being consumed. This establishes the source of funds and spending authority.

Outputs

Destination addresses and amounts. This defines where value moves and under what conditions it can be spent next.

Witnesses / Signatures

Cryptographic proof that the key holder authorized this specific transaction. This validates spending authority.

Flags & Lock Time

Control transaction malleability, replay protection, and time-based spending conditions. Misuse creates an exploit surface.

Blocks, Headers, and Tamper Evidence

The block header is the security manifest of each block. It binds historical state, transaction content, and consensus proof into a single cryptographically committed structure.

Block Header Fields

Previous Block Hash: links to parent block, forming the chain

Merkle Root: commits to all transactions in this block

Timestamp: approximate time of block creation

Version: protocol rules in effect for this block

Nonce / Difficulty: proof of computational work expended

Why This Matters for Tamper Evidence

Altering any transaction in a historical block changes its Merkle root, which changes that block's hash, which invalidates every subsequent block's "previous hash" field.

An attacker must redo all accumulated work from the altered block forward, making deep rewrites computationally prohibitive on large, active networks.

Why STRIDE Still Works

STRIDE is a structured threat modeling framework developed at Microsoft. Its six categories map cleanly onto blockchain attack patterns; even though the technology is novel, the threat categories are not.

Spoofing

Tampering

Repudiation

Information Disclosure

Denial of Service

Elevation of Privilege

The same threat category expresses differently at each layer of the stack. A defender trained in STRIDE can reason about blockchain attacks immediately: they just need to understand the new expressions.

STRIDE: Spoofing and Tampering

Spoofing in Blockchain

Stolen private keys: an attacker signs transactions indistinguishably from the legitimate owner.

Fake wallets: malicious software mimics legitimate wallet UI, capturing seed phrases and keys.

Malicious RPC endpoints: an attacker controls the node a client queries, feeding false state data.

Tampering in Blockchain

Transaction manipulation: mempool-level reordering or replacement attacks alter which transactions confirm.

State manipulation: exploiting smart contract logic to alter contract state in unauthorized ways.

Consensus-level rewrite: a 51% attacker can reorganize recent blocks, reversing confirmed transactions.

STRIDE: Repudiation and Information Disclosure

Repudiation

- Digital signatures provide strong attribution, but only to a key, not a legal person.
- Stolen keys allow an attacker to act under the victim's identity with no distinguishable signature.
- Key compromise creates plausible deniability for the attacker and genuine confusion for investigators.

Information Disclosure

Metadata leakage: IP addresses, timing, and transaction patterns can de-anonymize participants.

Private key leakage: Exposed keys reveal the full transaction history linked to that address.

Graph analysis: On-chain transaction flows can be analyzed to cluster addresses and trace fund movements.

- Public ledgers are permanently searchable; any future disclosure exposes all past activity.

STRIDE: DoS and Elevation of Privilege

Denial of Service

Node flooding: Overwhelming a node's network or processing capacity to knock it offline

Transaction spam: Flooding the mempool with low-value transactions to delay legitimate confirmations

Validator exhaustion: Resource-intensive operations designed to degrade consensus participant performance

Elevation of Privilege

Stolen admin keys: Compromise of a contract owner key grants full control over contract logic and funds

51% style control: Acquiring majority stake or hash rate grants the authority to reorder or censor transactions

Unauthorized contract access: Exploiting missing or incorrect access controls to call privileged functions

Apply STRIDE Across the Stack

STRIDE categories manifest differently at each layer; the same framework covers the full attack surface when applied with layer awareness.

Same Threat, Different Layer

Elevation of Privilege is a single STRIDE category; however, its expression, root cause, and required controls differ completely depending on where it occurs in the stack.

1

Key Layer

Key theft: attacker gains full signing authority. Control: secure key storage, HSMs, air-gapped signing.

2

Consensus Layer

Consensus takeover: majority stake or hashrate grants transaction ordering authority. Control: network diversity, slashing conditions.

3

Contract Layer

Admin bypass: missing access controls allow unauthorized privilege escalation. Control: formal verification, access control audits.

One STRIDE category requires three completely different defensive playbooks. Layer-aware threat modeling is not optional; it is the only way to drive actionable controls.

Threat Modeling Questions

Use these questions as a structured starting point whenever you are threat modeling a blockchain system, integration, or deployment. They cut through complexity to the security-critical variables.

1. ## What is the asset?

   Define what has value: funds, keys, contract state, governance rights, or data integrity. The threat model follows the asset.

2. ## Where is the trust boundary?

   Identify every point where the system must trust an external party, a piece of software, or a protocol assumption.

3. ## Who can authorize state change?

   Map every key, role, and mechanism that can alter the ledger or contract state. These are your highest-priority attack targets.

4. ## What validates the action?

   Understand what the network actually checks, and what it does not. Gaps between assumed and actual validation are exploit surfaces.

5. ## What happens if this layer fails?

   Map the blast radius. Layer failures in blockchain systems often cascade; a compromised consensus layer affects every application built on top.

Consensus as a Security Mechanism

Consensus is the protocol-level answer to a fundamental distributed systems problem: how do mutually distrusting participants agree on a single shared state without a central authority?

State Agreement

All honest nodes converge on the same valid chain history, preventing divergent or conflicting ledger views.

Byzantine Fault Tolerance

The network continues to function correctly even when a fraction of participants behave maliciously or arbitrarily.

Shared History

Consensus enforces a single canonical transaction history that becomes exponentially harder to rewrite as confirmations accumulate.

Integrity at Scale

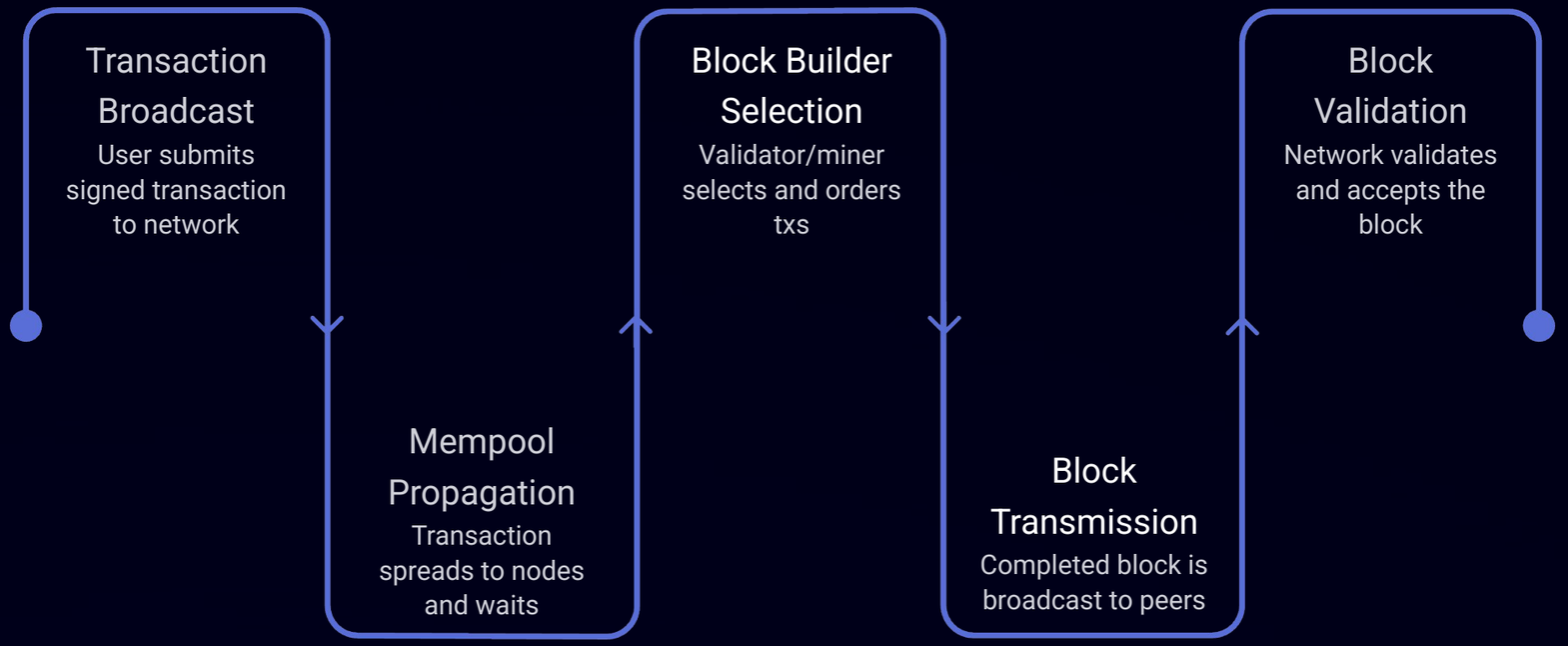
Without a central coordinator, consensus achieves ledger integrity across thousands of geographically distributed, mutually untrusted nodes.

PoW, PoS, and Security via Scarcity

Both major consensus models use a different form of scarcity to make attacks expensive. The security model, and therefore the attack surface, differs significantly between them.

Block Creation Is an Attack Surface

Every phase of the block creation pipeline is an opportunity for manipulation. The entity building and proposing a block has significant power over transaction ordering, inclusion, and censorship.



The block builder at Step 3 controls ordering, inclusion, and exclusion; these actions create opportunities for frontrunning, censorship, and MEV (Maximal Extractable Value) attacks.

Frontrunning and Ordering Risk

Transaction ordering is not neutral. In a public mempool, pending transactions are visible before confirmation, giving block producers and sophisticated observers an informational advantage they can monetize at other users' expense.

Pending Transaction Visibility

Transactions broadcast to the network sit in a public mempool before confirmation. Their contents, including size, destination, and intent, are visible to all observers.

Reordering for Advantage

Block producers can insert their own transactions ahead of a target transaction (frontrunning) or sandwich it with buy-and-sell transactions to extract value at the user's expense.

Inclusion and Censorship

Block producers can selectively exclude specific addresses or transactions: a form of censorship that undermines the permissionless property of public chains.

Integrity of Transaction Ordering

The assumption that transactions confirm in a fair, first-come-first-served order does not hold. Ordering is a security property that must be explicitly designed for.

Blockchain Nodes Are Still Just Computers

The elegance of the protocol does not change the fact that nodes are software running on operating systems, connected to networks, maintained by humans. All conventional attack surface applies.

Malware & Key Theft

Node operators store keys. Malware targeting the host OS can exfiltrate key material silently, bypassing protocol-level security entirely.

Stale Software

Unpatched node software carries known CVEs. Protocol-level security offers no protection against OS or dependency vulnerabilities.

Misconfiguration

Exposed RPC ports, weak authentication on admin interfaces, and permissive firewall rules are common in community-run node deployments.

Parser Bugs

Processing malformed blocks or transactions can trigger memory corruption, crashes, or consensus-splitting behavior in node software.

Network-Level Attacks

Blockchain networks are peer-to-peer systems built on standard internet infrastructure. Network-layer attacks can isolate nodes, partition the network, and undermine consensus without ever touching cryptographic primitives.

Eclipse attacks are particularly dangerous; a fully eclipsed node can be fed a false chain or be the target of double-spend attacks, and it has no way to detect the manipulation from within its isolated view.

Smart Contracts Expand the Attack Surface

Smart contracts are programs that run autonomously on the blockchain: they can hold funds, enforce rules, and execute logic without human intervention. That power creates a dramatically expanded attack surface.

What Makes Smart Contracts Different

Programmable logic: Arbitrary code executing on a shared, adversarial runtime

Asset-moving code: Contracts can hold and transfer significant value autonomously

Irreversible consequences: Exploits drain funds with no chargebacks, no reversals, and no appeal

Why Traditional AppSec Thinking Applies

- Input validation, access control, and state management flaws are all present.
- However, the execution environment introduces entirely new vulnerability classes with no equivalent in traditional software.
- Code is public: attackers can read the contract, find the bug, and craft the exploit before deploying it.
- Immutability means patching is often impossible without a proxy upgrade architecture.

Smart Contract Vulnerability Classes

Smart contract vulnerabilities span four categories. The most dangerous are blockchain-specific; they have no direct equivalent in traditional software security and require specialized knowledge to identify and remediate.

1

General Programming Flaws

Integer overflow/underflow, unchecked return values, improper input validation, and access control failures. These are familiar from traditional software, but they result in irreversible financial consequences on-chain.

2

Blockchain-Specific Flaws

Reentrancy attacks (such as the DAO hack), timestamp dependence, block number manipulation, transaction ordering attacks, and flash loan-enabled price manipulation.

3

Platform-Specific Flaws

EVM-specific quirks: delegatecall proxy vulnerabilities, storage collision in upgradeable contracts, gas griefing, and selfdestruct misuse.

4

Application-Specific Flaws

Protocol-level design errors: oracle manipulation, governance attacks via flash loans, bridge trust assumptions, and economic model exploits.

The App Layer Still Breaks

The protocol may be sound. The contracts may be audited. And yet, the user's experience of that system runs through a web browser, a mobile app, and a series of API calls: every one of which is a conventional attack surface.

Wallet Malware

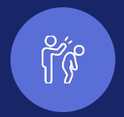
Clipboard hijackers silently replace copied wallet addresses. Keyloggers capture seed phrases at entry. Fake wallet apps harvest keys on install.

dApp Frontend Compromise

A compromised front-end JavaScript file can replace legitimate contract addresses with attacker-controlled ones, affecting all users without touching the contract itself.

Bridge Risk

Cross-chain bridges hold massive locked value and have been the target of the largest individual DeFi exploits to date, often via smart contract or key management failures.

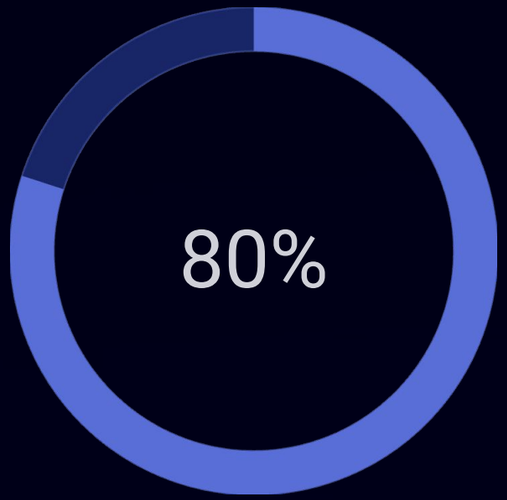


API Abuse & Session Hijack

Explorer APIs, wallet connect sessions, and browser extension permissions all represent conventional web attack surfaces with high-value consequences when exploited.

Most Real-World Losses Start with Familiar Attacks

Despite blockchain's cryptographic sophistication, the majority of documented financial losses have root causes that every security practitioner already knows, because attackers follow the path of least resistance.

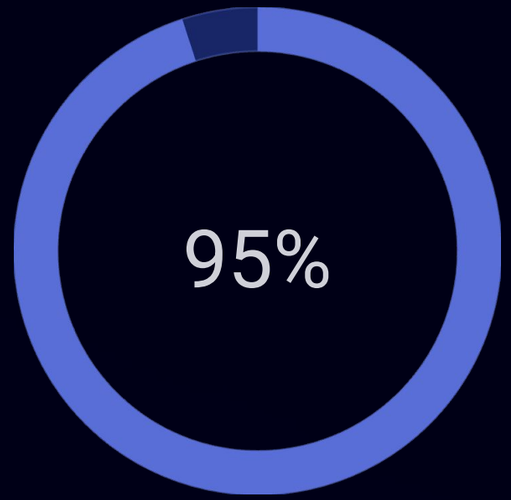


Of Major Losses

Attributed to application-layer attacks, such as phishing, malware, or key theft, rather than protocol-level failures.

Bridge Exploits

Cross-chain bridges account for a disproportionate share of total DeFi losses, often via compromised multisig keys or smart contract flaws.



Of Phishing Victims

Lose assets within minutes of key or seed phrase disclosure; blockchain transactions are irreversible and instant.

Credential stuffing, password guessing, SIM swapping, and hot wallet compromise are the dominant attack vectors. The blockchain is operating as designed; attackers target the humans and infrastructure around it.

Wallet and Exchange Operational Security

The security of a blockchain deployment is ultimately bounded by the operational security of the key management practices around it. These failures are systemic and well-documented.

Weak Key Storage

Private keys stored in plaintext files, environment variables, or unencrypted databases are one breach away from total loss. HSMs and secure enclaves exist for a reason.

Weak Recovery Paths

Seed phrases stored in cloud notes, email drafts, or screenshots create recovery paths that are far weaker than the key itself; attackers target these preferentially.

Poor MFA Design

SMS-based MFA for exchange accounts is vulnerable to SIM swapping. Hardware security keys and authenticator apps dramatically reduce this risk.

Bad Signing Hygiene

Signing transactions on internet-connected devices, blind-signing unverified contract calls, and approving unlimited token allowances are all high-risk behaviors with documented exploitation histories.

Designing Secure Blockchain Systems

Secure blockchain architecture requires deliberate choices across multiple dimensions. These are not implementation details: they are foundational design decisions that determine the system's threat model.

Architecture Choices

Public vs. Private: Open participation versus permissioned validator sets; these models create different trust requirements and attack surfaces.

Open vs. Permissioned: Who can read, write, and submit transactions? Each constraint changes the threat landscape.

Governance Design: Who controls protocol upgrades? Poorly designed governance is an escalation path.

Operational Considerations

Privacy Enhancements: Zero-knowledge proofs, confidential transactions, and mixing services all involve significant security trade-offs.

Secure Defaults: Minimal permissions, explicit access control, and upgrade path restrictions are vital.

Compliance Constraints: Regulatory requirements may mandate auditability features that conflict with privacy goals.

Final Takeaways

Blockchain Is Layered

Security exists at the cryptographic, data structure, protocol, infrastructure, and application layers: each with distinct attack surfaces and defensive controls.

Trust Is Shifted, Not Removed

You still trust something, such as code, mathematics, incentive structures, and the people who implement and operate them. Map those trust dependencies explicitly.

Keys Are the Highest-Value Secret

Private key compromise is game over. Key management is your highest-priority control surface in any blockchain deployment.

Most Losses Happen Around the Chain

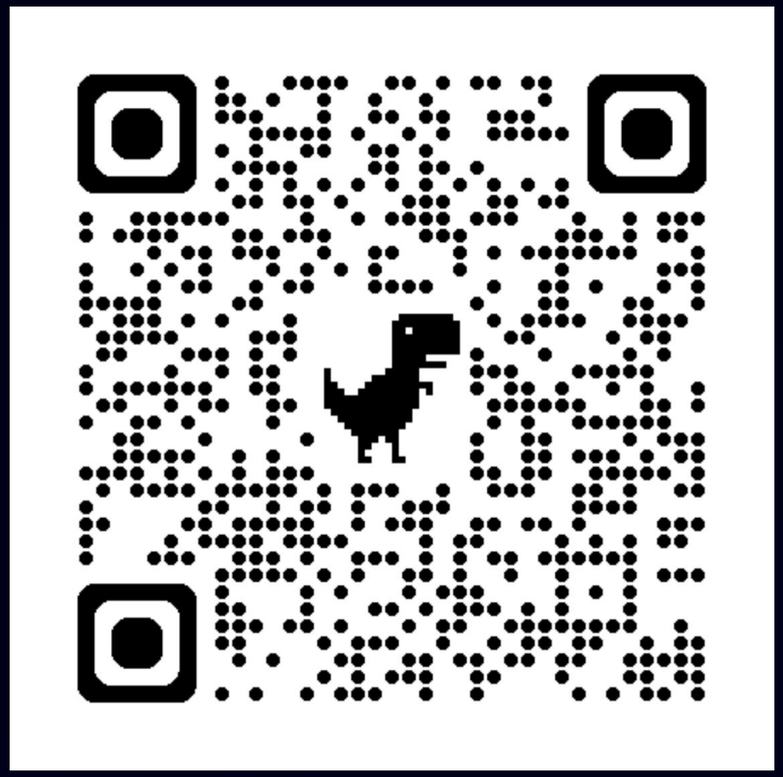
Phishing, malware, SIM swapping, and weak custody practices, rather than protocol exploits, dominate real-world loss events. Defenders own that terrain.

The full-stack view is not optional. Defenders who understand only the protocol layer are defending an incomplete model. The attack surface runs from the math to the user, and so must your threat model.

Michael "0xObsidianEnoch" Lathan | NebraskaCERT Lunch Talk

Let's Connect

Michael "0xObsidianEnoch" Lathan Jr.



Scan to find me on the web

No, I am not hacking you. This is a legitimate QR code for my contact info and socials. You can scan it safely. 😄