Attacchi sulla Blockchain e Discussione su Wallet con Standard Emergenti

Attacchi al Livello di Smart Contract e Applicazioni

Reentrancy Avanzata e Varianti

L'attacco di reentrancy ("The DAO" su Ethereum) è già noto, ma col tempo sono emerse **varianti più complesse**:

1. **Reentrancy multi-step**: l'attaccante chiama una funzione che, a sua volta, effettua una call esterna e poi torna in un punto differente del contratto – magari sfruttando sub-funzioni con "delegatecall."

2. **Cross-Contract Reentrancy**: se un dApp A chiama B, e B richiama A su un'altra funzione, si crea un reentrancy "circolare."

3. **Pull vs Push Payment**: Le best practice suggeriscono di "**pull**" i fondi (il beneficiario chiama withdraw) invece di "**push**" (il contratto manda i fondi in anticipo). Ci sono contratti che non rispettano questo pattern, diventando vulnerabili.

In generale, la reentrancy colpisce le logiche di **aggiornamento di stato** non protette da un meccanismo di lock o dall'ordine "Checks-Effects-Interactions." Un attaccante abile può **ripetere** la chiamata prima che il saldo sia aggiornato, sottraendo risorse multiple volte.

Soluzione:

- Pattern "Checks-Effects-Interactions"
- Uso di **ReentrancyGuard** (OpenZeppelin)
- Minimizzare le call esterne a contratti non affidabili

Oracle Manipulation e Price Feeds

Molte dApp DeFi si basano su **oracoli** che forniscono dati di prezzo (ad es. ETH/USD) o informazioni esterne. Se l'attaccante controlla (o influenza) l'oracolo, può manipolare i prezzi, generando **profitti illeciti**:

- **Esempio**: in un DEX decentralizzato, se il prezzo di un token X è basato su un oracolo, l'attaccante può gonfiare artificialmente il prezzo di X, farsi prestare stablecoin in eccesso e scappare con i fondi.

- **Attacco flash loan**: l'attaccante prende in prestito flash loan, manipola il pool di liquidità con un "price oracle on-chain," e ripaga il flash loan nella stessa transazione, intascando i guadagni.

Soluzione:

- Oracoli **decentralizzati** (Chainlink, Band) con feed multipli e meccanismi di aggregazione.

- Accordi di "time-weighted average price" (TWAP) che riducono i picchi momentanei.

- Sistemi di soglia e reputazione: l'update oracolo deve passare da un comitato di nodi non collusi

Front-Running e Maximal Extractable Value (MEV)

In alcune chain (soprattutto EVM-based), i validatori/miner **vedono** le transazioni in mempool prima di confermarle, e possono **riordinarle** a proprio vantaggio. Queste operazioni si chiamano front-running, back-running o sandwich attack, e fanno parte del fenomeno del **MEV (Maximal Extractable Value)**.

1. **Front-running**: se un utente vuole comprare un token con 1 ETH e lo annuncia nella mempool, un validatore può inserire la propria transazione **prima** per comprare il token a un prezzo più basso e rivendere subito dopo a un prezzo più alto, guadagnando.

2. **Sandwich Attack**: Il validatore “infiltra” una buy e una sell attorno alla transazione vittima, guadagnando sulla variazione di prezzo creata dal trade della vittima.

Impatti: Per l’utente, la transazione risulta molto più costosa, o fallisce, e i validatori ottengono guadagni extra. Sulle chain con meccanismi di “order-based mempool,” è un problema notevole di design.

Soluzioni:

- Protocollo **Flashbots** (bandisce l’ordine trasparente, propone un meccanismo di “private transactions” con righe dedicate).
- Modelli di transazioni private (Aztec, Secret Network, etc.) per nascondere i parametri dell’ordine.
- “MEV Auctions” o “MEV smoothing” proposti in Ethereum 2.0, per ridurre la manipolazione.

Attacchi di Governance e Manipolazioni Interne

3.1 Hard Fork e Social Attacchi

La **governance** di una blockchain permissionless non è soltanto “on-chain,” ma spesso “social.” Se un attaccante possiede grande influenza (o stake), potrebbe:

- **Indurre** un hard fork “a sorpresa,” convincendo miner/validator a seguire la sua chain, invalidando transazioni sgradite.

- **Minacciare** di vendere massicciamente il token per causare crash di prezzo, forzando la community ad adottare certe proposte di governance.

Non si tratta di un attacco “tradizionale,” ma di un manipolo di attori col potere di creare un “network split” (un fork) o minacciare la stabilità economica.

Soluzione: Meccanismi di governance con maggioranze qualificate, trasparenza nelle proposte (EIP/BIP su Ethereum/Bitcoin), e lock di token che impediscano vendite istantanee.

Upgrade Attack

In alcune chain, esiste una logica di **upgrade** dei contratti o del protocollo. Se l'attaccante riesce a proporre un upgrade malevolo e controlla la maggioranza di voto (in PoS) o la fiducia (in blockchain consortili), potrebbe introdurre una backdoor, alterare la logica di validazione o manipolare i parametri.

Esempio: Un "governance contract" su un DEX DeFi in cui la chiave di upgrade è gestita da un team. Se un attaccante acquisisce la chiave (o la corrompe dall'interno), può rimpiazzare il contratto con una versione che ruba i fondi degli utenti, "in modo legittimo."

Soluzione:

- Procedure multi-sig e di time-lock: l'upgrade dev'essere annunciato e rimandato di X giorni, per dare modo alla community di reagire.

- Auditing del nuovo codice e threshold signatures su governance.

Attacchi a Cross-Chain bridging e Interoperabilità

4.1 Manipolare i Bridge

Quando due blockchain differenti (es. Ethereum e un layer-2) si collegano via “bridge,” si blocca un asset su una chain e si mintano token corrispondenti su un’altra. Un attaccante che riesce a:

1. Falsificare le prove di blocco (“light client” compromised)
2. Prendere controllo dei validatori del bridge federato
3. Trovare un bug di smart contract bridging

può “sboccare” asset su una catena, senza un vero lock sulla catena originale, creando duplicazione. **Esempi**: Wormhole Hack (2022), Ronin Hack (600+ milioni), Nomad bridging.

Soluzioni:

- Bridge con meccanismi multipli di validazione (M-of-N relayer, BFT-based bridging).
- Formal verification dei contratti di bridging, auditing approfondito.
- Design “light client” su chain B che convalida in modo sicuro la finalità su chain A (richiede un protocollo ben testato e costoso da eseguire).

Side-Channel e Attacchi a Livello di Infrastruttura

5.1 Side-Channel (CPU, Timing, Caches)

In alcuni contesti blockchain, specialmente su enclavi di sicurezza (SGX) o su layer-2, un attaccante potrebbe sfruttare canali laterali (variazioni di tempo di risposta, pattern di memoria cache) per estrarre chiavi o dati interni. Sebbene meno comune su catene pubbliche, è rilevante in soluzioni di “trusted execution environment” e in blockchain permissioned enterprise.

5.2 DNS Hijacking e DevOps Attack

Anche se la blockchain “on-chain” è sicura, un attaccante può colpire l’infrastruttura off-chain, ad esempio:

- **DNS hijack** di un explorer o di un provider di wallet, reindirizzando gli utenti a un sito phishing.
- **DevOps supply chain**: se un team di dev di un protocollo DeFi subisce un attacco, un update malevolo può rubare fondi dei contratti.
- **Node software**: un repository GitHub corrotto potrebbe distribuire un client con backdoor.

In tali casi, la sicurezza della blockchain deve estendersi a procedure di firma e controllo integrità (ad es. NIST supply chain guidelines, code signing, polizze di devOps zero-trust).

DISCUSSIONE APPROFONDITA SUI WALLET E STANDARD EMERGENTI (NIST 2.0)

Wallet e Sicurezza

6.1 Evoluzione dei Wallet

Un **wallet** è l'interfaccia con cui l'utente gestisce le chiavi private e interagisce con la blockchain. Nel tempo, i wallet sono passati da:

1. **Paper wallet** (stampa della chiave)
2. **Desktop wallet** (chiave salvata in un file)
3. **Hardware wallet** (Ledger, Trezor) con chip sicuro
4. **Smart contract wallet** (multisig, social recovery, MPC)

Ciascun tipo affronta rischi di phishing, keylogger, furto di device, mancanza di backup, ingegneria sociale, ecc. I protocolli emergenti spingono su:

- **MFA** (Multi-Factor Authentication)
- **Threshold signature** (2 su 3)
- **Social Recovery** (un gruppo di contatti che possono ripristinare la chiave)

Principi NIST Cybersecurity Framework 2.0

Il **NIST Cybersecurity Framework** (CSF) definisce 5 funzioni: **Identify, Protect, Detect, Respond, Recover**. Applicato ai wallet:

1. **Identify**: Mappare gli asset digitali (chiavi, seed phrase) e le minacce potenziali (phishing, malware, bridging malevolo).
2. **Protect**: Uso di passphrase robusta, encryption a riposo, hardware wallet, procedure di backup.
3. **Detect**: Rilevare tentativi di accesso non autorizzati (log di accesso, notif push su telefono), analisi comportamentale.
4. **Respond**: Se si subisce una compromissione parziale (es. un “relayer” in un wallet MPC), revocare la share, rigenerare la seed.
5. **Recover**: Pianificare la “recovery phrase” offline, social recovery, procedure di emergenza per reimpostare un wallet su un device diverso.

La **versione 2.0** del NIST CSF, in elaborazione, aggiorna linee guida su supply chain, sicurezza di ambienti cloud e orchestrazioni. Per un wallet, si raccomanda:

- **Gestione robusta delle chiavi** (FIPS 140-3 per moduli criptografici, PKI)
- **Access control** multi-layer (MFA, device-binding)
- **Procedure di auditing** e logging su ogni transazione firmata.

Standard Emergenti e Tipologie di Wallet Moderni

Hardware Wallet con Secure Element e FIDO2

- **Ledger** e **Trezor** usano un **secure element** (chip) con architettura di sicurezza.
- L'utente sblocca con PIN, e il dispositivo firma le transazioni offline.
- **FIDO2** ha definito standard di key attestation: alcune implementazioni pensano a un wallet che funzioni come un token USB di sicurezza.

Smart Contract Wallet (Argent, Gnosis Safe, etc.)

- **Multisig**: un wallet on-chain richiede la firma di M-of-N chiavi. Nessuna singola compromissione blocca i fondi.
- **Daily limit** e "Guardians": L'utente impone soglie di spesa al giorno e nominano guardiani (amici) per sbloccare il wallet se la chiave è persa.
- **Recovery**: Se la chiave principale è compromessa, i guardiani (o un meccanismo MPC) possono rigenerare i permessi.

MPC Wallet e Social Recovery

Alcune piattaforme adottano protocolli di **MPC** (Secure Multi-Party Computation). La chiave privata non esiste mai interamente in un singolo dispositivo: viene "spezzata" tra più server/parti. Per firmare, si esegue un protocollo crittografico dove ciascuno calcola la propria parte e ottiene la firma ECDSA finale.

- Se un server è corrotto, non basta per rubare i fondi.
- Si possono definire policy (ad es. soglia di firme, orari consentiti) adatte a contesti enterprise.

Social recovery: L'utente "condivide frammenti" della seed con 2-3 amici di fiducia, un meccanismo che NIST definirebbe come un "privileged control group." Se l'utente perde la chiave, bastano i frammenti di almeno 2 amici per ricostruirla.

Esempi Finali di Wallet con Standard Emergenti e Best Practice NIST

1. **Ledger Nano + 2FA**: L’utente sblocca un hardware wallet con PIN e poi conferma via un’app “TOTP” (time-based one-time password). Il firmware segue linee di sicurezza FIPS.

2. **Gnosis Safe** (Ethereum) con “n-of-m signers,” definendo soglie su spese massime/giorno. Se un attaccante corrompe una chiave, serve comunque la maggioranza.

3. **MPC in un’azienda**: 3 manager custodiscono frammenti. Per spese di routine, basta 2 firme; per spese over 100.000 USD, ne servono 3. I log vengono inviati a un SIEM (Security Information and Event Management) per detect e respond, in linea con NIST.

4. **Social Recovery**: “Argent wallet” su Ethereum, con “guardians” (es. amici e family) per resettare la chiave se persa.