Bitcoin attacks

April 29, 2023

Good afternoon my Alices. So let's go to the wonderful world of bitcoins♠☺

Contents

1 Cryptography .

First, let's talk about cryptography, because blockchain technology is based on cryptography.

- **ECC (Public Key Cryptography)** is a technology often used to authenticate data using asymmetric encryption. 1 ECC was first used primarily to encrypt and decrypt messages in traditional computing.

- Cryptocurrencies now use this technology to encrypt and decrypt transactions. The key to ECC is "loophole functions," one-way mathematical functions that are easy to solve in one way, but almost impossible to reverse. Although both public and private keys are designed to protect a transaction, they differ significantly in their purposes. When they are compared to each other, the public key is used to verify the transaction after the transaction has been requested.

- **Public Key** - Usually a public key also translates to an "address" to receive cryptocurrency. Whereas the private key associated with an account is intended to authorize the transaction.

- **Private Key**- Generally, the private key is not disclosed and only the owner needs to know it. In other words, if someone gains access to your private key, they will have the right to deplete your assets in your wallet. As we know, we use a key pair.

Both of these keys are generated using elliptic curve cryptography. First, ECC creates a private key and then creates a public key from the private key using the elliptic curve algorithm (also known as ECDSA). Consequently, both the private key and the public key are cryptographically and mathematically related.

Generally ECC cryptography is considered to be the natural modern successor of RSA cryptosystem (I don't like it, it is too much cve, it needs a separate post to discuss its shortcomings) because ECC uses less keys and signatures than RSA for the same level of security, plus it provides instant key generation, key negotiation and quick signatures.

1.1 I will tell you how the key is generated.

Now let's look at how the key is generated.

- Any number within a range is a valid ECC private key.

- The public keys in ECC are EC points - pairs of integer coordinates x, y, lying on a curve. Due to their special properties, EC points can be compressed to just one coordinate + 1 bit (odd or even).

- Thus, a compressed public key corresponding to a 256-bit ECC private key is a 257-bit integer. In this format, the public key actually takes up 33 bytes (66 hexadecimal digits), which can be optimized to exactly 257 bits.

Algorithms for this encryption can use different basic elliptic curves, and different curves provide different levels of security, performance/speed, and key length, and to all this may include different algorithms.

The ECC curves adopted by popular cryptographic libraries and security standards have a name (named curves), such as "$secp256k1$" or "$Curve25519$", a field size that determines the key length, (for example 256 bits), the security

level (usually field size / 2 or less), performance (operations/sec) and other parameters. ECC keys have a length, which directly depends on the base curve.

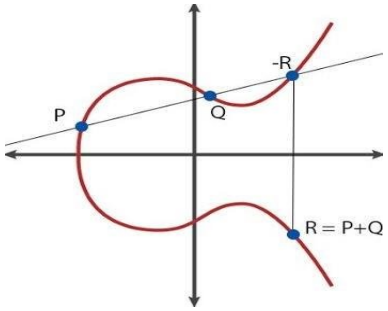
In most applications, such as OpenSSL, OpenSSH and Bitcoin, the default key length for ECC private keys is 256 bits, but depending on a curve, different ECC key sizes are possible: 192-bit (curve "$secp192r1$"), 233-bit (curve "$secp233k1$"), 384-bit (curves "p384" and "$secp384r1$"), etc.

Bitcoin uses "$secp256k1$", which is based on the formula:

$$y2 = x3 + 7$$

The equation of the elliptic curve itself is not very different:

$$y2 = x3 + ax + b$$



a and b are any arbitrary constants. For example, our favorite "$secp256k1$" uses$a = 0$, $b = 7$

1.1.1 Elliptic curve properties.

Elliptic curves have different properties.

- Thus, elliptic curves are symmetric about the x.
- For each point B which lies on an elliptic curve we can obtain its reflection .
- Or a non-vertical line intersecting the curve at two points will always intersect a third point on the curve.
- Since this curve is defined over a finite field of simple order, not over real numbers, it looks like a set of points scattered in two dimensions, which makes visualization difficult, but the math is identical to an elliptic curve over real numbers.

1.1.2 It's hard for many people to understand what we're talking about, but let's continue on our journey.

Given two points $P1$ and $P2$ on an elliptic curve, there is also a third point $P3$, which is equal to $P1 + P2$.

As we remember from their properties above geometrically this third point $P3$ is calculated by drawing a line between $P1$ and $P2$, so this line will intersect the elliptic curve in exactly one additional place.

ECC uses elliptic curves $F p$ (where p is a prime number and $p > 3$) or $F 2m$ (where field size$p = 2m$).

This means that the field is a square matrix of size pxp

$$\begin{matrix} {}_1p_1 & {}_1p_p \\ {}_pp_1 & {}_pp_p \end{matrix}$$

And points on a curve are bounded by integer coordinates only within the field. All algebraic operations in the field (addition and multiplication of points) lead to another point in the field.

The equation of an elliptic curve over a finite field $F p$ takes the following modular form:

$$y2 \equiv x3 + ax + b(modp)$$

For those who are poorly versed in cryptography, let me explain that mod returns the remainder of division of the number by the divisor $38(mod5) = 3(38 : 5 = 7, 6; 5 * 7 = 35; 38 - 35 = 3)$, $26(mod5) = 1$, there is also div integer from $5(div2) = 2(5 : 2 = 2, 5$ take $2)$, $26(div5) = 5$

Accordingly, the bitcoin curve "$secp256k1$" takes the following form:

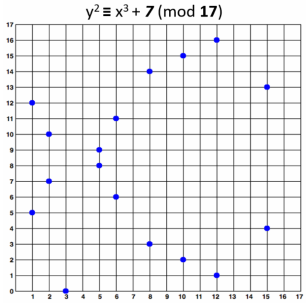
$$y2 \equiv x3 + 7(modp)$$

An elliptic curve over a finite field Fp\digamma p consists of a set of integer coordinates x, y, such that $0 \leq x$, $y < p$, remaining on the elliptic curve:

$$y2- \equiv x3 + ax + b(modp)$$

Example of an elliptic curve over a finite field:

$$F17 : \mathrm{y2} \equiv \mathrm{x3} + 7(mod17)$$



In practice, elliptic curves used in cryptography are sets of points in a square matrix, not classical "*curves*". It is fairly easy to calculate whether a given point belongs to a certain elliptic curve over a finite field. For example, a point x, y belongs to a curve:

$$y2 \equiv x3 + 7(mod17)4$$

Only when:$x3 + 7 - y2 \equiv 0(mod17)$ The point $P4$, belongs to the curve, since:

$$(4 * 3 + 7 - 7 * 2) : 17! = 0$$

1.1.3 Key Genination .

The first and most important step in creating keys is to find a safe source of entropy or randomness.

The private key can be any number from 1 to $(n - 1)$, where n is a simple constant denoting the order of the base point G.

Hex and raw binary formats are stored for software use and are not normally shown to the end user.

In python, the $hex()$ method converts an integer to its hexadecimal representation and outputs a string of the form ' $0x$....', where the prefix $'0x'$ refers to the hexadecimal format.

It is possible that the randomly generated private key will not be large enough to fill all 256 bits, in which case we need to add enough leading 0's to make the final length equal to 256 bits. The exact method you use to select this number is irrelevant unless it is predictable or repeatable.

The Bitcoin software uses the base operating system's random number generators to create 256-bit entropy or in simple terms, randomness.

Usually the OS random number generator is initialized by a human random source, so you may be asked to move the mouse for a few seconds, more precisely the private key can be any number between 1 and $n - 1$, where n is a constant ($n = 1,158 * 1077$, just under 2256) defined as the order of the elliptic curve used in bitcoin.

To create such a key, we randomly choose a 256-bit number and check that it is less than n-1.

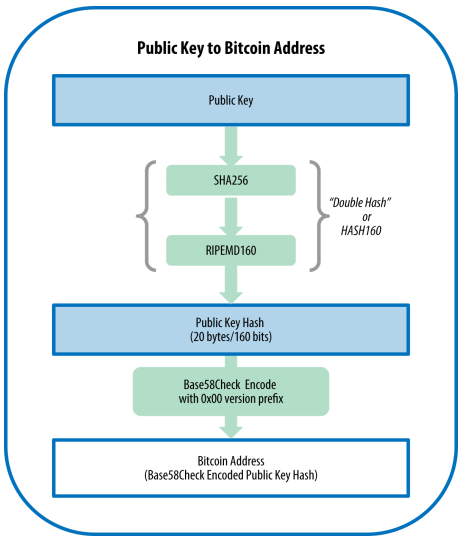
In programming terms, this is achieved by feeding a larger string of random bits, collected from a cryptographically secure random source, into the "$SHA256$" hash algorithm, which conveniently creates a 256-bit number.

Accordingly, if the result is less than $n - 1$, we have a suitable private key, otherwise we just try again with a different random number.

The public key is computed from the private key using elliptic curve multiplication, which is irreversible: where k is the private key, G is a constant point called the generator point, and K is the resulting public key.

- The reverse operation, known as finding the discrete logarithm - calculating k if you know K - is as complicated as trying all possible values of k, or as brute-force searching.
- The bitcoin address is obtained from the public key using one-way cryptographic hashing.
- The hashing algorithm is a one-way function that creates a fingerprint or "$hash$" of arbitrary size input.

Bitcoin addresses are almost always presented to users in an encoding called "$Base58Check$":



1. Inserts a prefix at the beginning of the original private key
2. $Sha256$ is performed (I think twice) for binary representation of the new prefix key.
3. The first 4 bytes (8 half bytes) are stored in the checksum variable.
4. The checksum is added to the end of the prefix key.

1.2 Blockchain .

There are also various scripts used to manage the transfer of an asset from payer to recipient in the bitcoin network, the standard types of scripts are:

$$Pay - to - Public - Key - Hash(P2PKH)$$

It is a form of cryptocurrency that is mostly used in bitcoin to make transactions.

P2PKH cryptocurrency is computed by sending a public key and a digital signature made by the corresponding private key.

Other scripts used in bitcoin transactions include the following: $Pay - to - Script - Hash(P2SH), Pay - to - MultiSig(P2MultSig), Pay - to - MultiSig(P2MultSig), Pay - to - PublicKey(P2PK)$.

1.3 Additional Materials .

I hope you were able to grasp the basic theory about elliptic curves, because it will be more complicated.

Additional material will be in the archive below, also here is a link to information about bitcoin **key generation in it**.

And now I will describe some useful articles for learning cryptography itself.

1. **A small but useful practical article on cryptography**
2. **research in cryptography during the course "ECE579C - Applied Cryptography and Physical Attacks", the essence of the research is to write a balanced Simon cipher coding on an ARM embedded processor. Cryptanalysis and cryptographic attacks on various algorithms**
3. **Examples of some crypto algorithms in Python**
4. **Repository containing examples of Sage and/or Python attacks on popular ciphers**
5. **A collection about coding theory in cryptography**
6. **Here you will find a listing of the rules for writing code to implement cryptographic operations**
7. **A cool course of video lectures about cryptographic attacks, you can open the author's channel and view other useful materials**

8. **Collection of lectures and assignments on cryptography**
9. **Book, for those who want to understand elliptic curves**
10. **A book to help understand elliptic curves**
11. **A book about simulation groups which will help to understand elliptic curves**
12. **if you are interested in the structure and security of web3 you can read a cool article of my friend where everything is explained in a simple form**

1.3.1 Let's move on.

A Bitcoin transaction is a cryptographically signed statement that transfers the exact amount of coins from the sender's address to the recipient's address.

For the transaction, the payer proves ownership of the bitcoin amount with his private key (it is already recorded in the ledger), which transfers the bitcoin amount to his address.

The recipient uses his key pair to get the bitcoin amount to his address, and as we remember the bitcoin address is formed from the public part of the "$ECDSA$" public/private key pair with the curve name "$secp256k1$".

The client can sign the transaction with his private key, so anyone who knows the hex hash of the Bitcoin address public key can verify that the signature is valid.

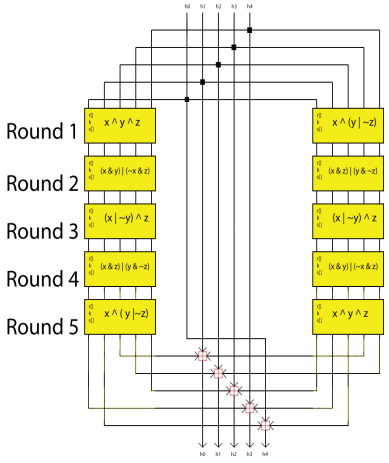
$$constsum = "SHA256"(SHA256(prefixhash))[0...3]$$

$$address = "Base58"(prefixhashchecksum)$$

A one-way cryptographic secure hash function is computed along with "$RIPEMD$" to generate a bitcoin address, given the elliptic curve domain parameter (p, a, b, G, n) and the user key pair $(d, P = dG)$, then the address is generated by the application:

$$hash = RIPEMD160(SHA256(ECPublickey))$$

1.4 Let me talk briefly about RIPEMD.



Bitcoin uses $RIPEMD-160$.

- $RIPEMD-160$ is a fast cryptographic hash function oriented on software implementations of 32-bit architectures and evolved from the 256-bit $MD4$ extension.
- The main feature of $RIPEMD-160$ are two different and independent parallel chains, the results of which are combined at the end of each application of the compression function.
- It should also be noted that $RIPEMD-160$ provides 160-bit results and is designed to provide a high level of security.
- RIPEMD-160 compresses an arbitrary sized input string into blocks of 512 bits each, then each block is divided into 16 byte strings and the 4 byte string is converted into a 32 bit word using the "$little-endian$" convention (where junior to senior means that the record starts with junior and ends with senior) used in the $Intel80x86$; $MD4/MD5$ and $RIPEMD$ architecture.
- The result of $RIPEMD-160$ is contained in five 32-bit words which form the internal state of the algorithm. The final contents of the five 32-bit words are converted to a 160-bit string, again using the $little-endian$ convention.
- This state is initialized with a fixed set of five, 32-bit words with an initial value, and the main part of the algorithm is known as the compression function (it makes up the new state from the old one and the next block of 16 words).
- The compression function consists of five parallel steps, each with 16 steps.

$$5 * 16 * 2 = 160$$

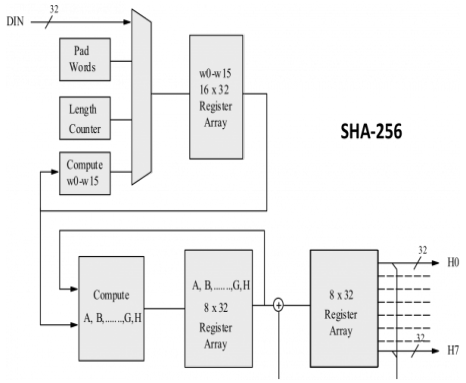
Compared to :

$$3 * 16 = 48$$

For MD4. Initially two copies are made from the old state (five left and right registers of 32 bits each), both halves are processed independently.

- Each step calculates a new value for one of the registers based on the other four registers and one message word. At the end of the compression function we calculate the new state by adding to each word of the old state one register from the left half and one from the right half.

1.5 Let me tell you about SHA 256.



$SHA256$ is part of the "$SHA2$" family of algorithms, where SHA stands for Secure Hash Algorithm and the number 256 in the name stands for the final value of the hash, i.e. no matter the size of the opened/open text, the hash function value will always be 256 bits.

You can divide the entire encryption process into five different segments that add a few extra bits to the message, so that its length is exactly 64 bits less than a number multiple of 512.

During the addition, the first bit must be one, and the rest of it must be filled with zeros.

Now you can add 64 bits of data to make the final open text a multiple of 512.

You can calculate these 64 character bits by applying the module to the original open text without filling, you need to initialize the default values for the eight buffers that will be used in the steps.

You must also store 64 different keys in the array, ranging from $K[0]$до$K[63]$.

All messages are divided into several blocks of 512 bits each. Each block goes through 64 cycles of operations, with the output of the blocks serving as input for the next block, while the value of $K[i]$ is pre-initialized in all these steps,

$W[i]$ is another input that is calculated individually for each block depending on the number of iterations being processed at the time.

At each iteration, the final block result serves as input for the next block, and the whole cycle is repeated until you reach the last 512-bit block, after which its output is considered a final hash, which will be 256 bits long.

2 Phishing attacks.

Now let's talk about the attacks themselves, the first thing I will talk about is phishing with spoofing wallet addresses 9.

2.1 Phishing with spoofing wallet addresses .

The first thing I will talk about is phishing with spoofing wallet addresses.

2.1.1 Phishing for bitcoin address spoofing, without access to the victim's PC.

1. First, the attacker is looking for active, so-called "*not dead*" wallets, as an example, take the **site**, where by selecting a suitable wallet the attacker chooses a target.

1. Next, the attacker generates an address that looks similar to the address of the potential victim. For his purposes, the attacker can use the wallet generation website "*bitcoin-generator-2018.b*", or other tools, of which there are plenty in the network. Don't forget that wallet addresses consist of randomly generated strings of alphanumeric characters, of course, depending on the blockchain, they can have a length of 25 to 40 characters (not including the prefix). Providers use abbreviated versions that display only the first and last few characters in the address to make it easier to handle. In an "*Address Poisoning*" attack (a type of cyberattack that exploits weaknesses in the widespread Address Resolution Protocol (ARP) to disrupt or redirect or track network traffic.) attackers poison victims' transaction histories by using similar addresses with the same short form (i.e. with the same first and last characters).

2. The attacker sends the user a small amount of bitcoins or even 0. When users send or receive cryptocurrency, it appears in the wallet's transaction list. Clicking on the transaction displays additional information, including the token, the amount sent or received, and a brief form of the third party's address. This attack assumes that the user does not know his or her address, and not everyone can remember their wallet address consisting of a set of random numbers and letters. This method effectively obfuscates the transaction history with multiple entries that look like they are between the same address, but use different ones - one address for an actual legitimate transaction, and one from an attacker using a copycat wallet address.

3. Next, the attacker is waiting and hoping that the user will send the crypto-volume to the wallet stored in his history, but instead he uses the "*copycat wallet*" of the scammer and so to say without realizing it himself will transfer the cryptocurrency to the attacker.

To be honest, it is not a very good option, but as the saying goes,

the fish doesn't eat the fish. As an example, I suggest you look at information about a similar type of phishing

2.1.2 Phishing for bitcoin address spoofing, with access to the victim's PC.

Many of us have heard about stories of potential victims copying their wallet address using hotkeys "$ctrl+c$", "$ctrl+v$" in the form filling line and entering an attacker's address, i.e. the recipient's cryptocurrency wallet address is replaced with the address belonging to the attacker.

This tool demonstrates how some attackers redirect cryptocurrency transactions by replacing data from the clipboard, so when users copy the addresses of cryptocurrency wallets to which they want to transfer bitcoins, the copied information is imperceptibly replaced with the attacker's information.

Let's figure out how it works:

1. The first thing an attacker does is to drop a virus on the potential victim, and if he has physical access to the device (PC), he simply downloads it
2. Next, the potential victim copies the address of his bitcoin wallet, in full confidence that the address is correct;
3. The intruder's program checks the buffer for the presence of the needed symbols, indicating that this is a crypto-purse;
4. If it is confirmed that the target is detected (crypto-purse), the attacker's program replaces it with the address prepared before this operation;

Another crypto-malware that has been around since 2017 is known as "Clipper. It works by using the user's clipboard to replace the destination crypto address with the attacker's address.

For example, a user can easily copy and paste the address while trying to pay a friend, but they are actually transferring funds to an attacker. Since the crypto-address is usually very long, it is easy to confuse your real address, even if you try to match the address character by character. By the way, the Clipper program has been spotted on Bitcoin and Etherium platforms.

Examples of tools:

1. BTC-Clipper
2. Crypto-Clipper
3. Bitcoin-Clipper
4. Bitcoin-Clipper
5. BTC_address_changer
6. BTC-Clipper

2.2 Fake Browser Extensions .

Now, I'd like to talk about a simpler but common phishing attack.

A well-known entry among common crypto phishing attacks refers to fake browser extensions.

Cryptocurrency users use different types of browser extensions, as well as MetaMask wallets or other, similar types of cryptocurrency wallets.

While browser wallet extensions do provide some convenience for cryptocurrency users, this type of extension can be an easy target for cybercriminals.

Cyber criminals use fake cryptocurrency browser extensions to trick users, and fake browser extensions can help to obtain credentials to log into a user's wallet.

One example of such an attack hit the newswire last year, when a similar extension was downloaded more than 120 times on the Chrome Web Store.

A malicious Chrome extension called Ledger Live used Google Ads to promote the extension with a sense of legitimacy.

The confusion over original and fake browser extensions requires an answer to the question of how to avoid such crypto-fishing attacks.

And, interestingly enough, you can avoid fake browser extensions if you are more vigilant.

Rules for the Vigilant, or how not to lose digital money:

1. Never trust online stores when looking for crypto-extensions, but instead, do your best to check the crypto-extension profile page.

2. Review the extension profile page to find genuine reviews and the web extension development team.

3. You can find out if the extension is genuine if the reviews and the identity of the development team are genuine.

4. Most importantly, you should also analyze the information you get and understand for yourself that the extension is indeed genuine.

5. If you find any discrepancies between the permissions required for the extension and its advertised features, delete immediately and do not go over, do not download the extension.

6. Another simple solution to see if it's genuine is to download it from the developer's official website.

2.2.1 Phishing Bots .

The last and most interesting entry among popular crypto scams is phishing bots. Naturally, phishing bots have been used in the past to perform various kinds of tasks, but the use of phishing bots in cryptocurrency is primarily aimed at compromising valuable information for both the user and the attacker.

By the way, MetaMask told users in a statement that a group of bots intercepting Twitter phrases was responsible for such a phishing attack:

> Apparently, the phishing attack comes from an account that looks almost identical to the MetaMask account.
>
> The request requires filling out a support form on large sites such as Google Sheets, or searching for a secret user recovery phrase.
>
> It's an interesting fact, but the ingenuity of phishing bots can be very sophisticated.

Many of you have thought about the fact that you can check the authenticity of a message on an official account for example, and this is a good idea, because you can avoid a phishing attack by checking whether the message came from an official website or a phishing page.

But we are well aware that it is almost impossible to instantly determine if an official page or a web resource is hacked, because many social media accounts have been hacked in the past, and the most prominent example is Twitter, which was hacked in 2020:

> This hack, considered one of the most insidious crypto-attacks, led to the theft of bitcoins totaling $121,000.

In addition to these common examples of cryptocurrency phishing attacks, I want the reader to be aware of the existence of other attacks not described in the article, so you have to be careful, because phishing can be different and there can be many of them. So be careful and avoid such attacks.

3 Vulnerabilities of wallet applications .

3.1 Introduction.

Since we mentioned phishing in applications, let's talk about vulnerabilities of wallet applications, because they are not so safe either, especially when there is physical access.

Bitcoin is currently the most popular cryptocurrency. With the proliferation of smartphones and high-speed mobile Internet, more and more users are accessing their Bitcoin wallets on smartphones.

Users can install various bitcoin wallet applications (Coinbase, Luno, Bitcoin Wallet) and access them anytime, anywhere. Nevertheless, many bitcoin wallets are quite vulnerable now I will give you some half-baked vulnerabilities and you will see it.

There are several potential security risks associated with using a cryptocurrency wallet.

One risk is that attackers can gain access to the wallet through vulnerabilities in the software or application.

This can happen if the wallet is a "hot" wallet, meaning it is connected to the Internet and accessible through a mobile device or personal computer.

Some wallet apps may also have features that allow you to export private keys, which can increase the risk of theft if misused.

Another risk is the possibility of losing access to the wallet if login credentials are forgotten or the cell phone linked to the account is lost or stolen.

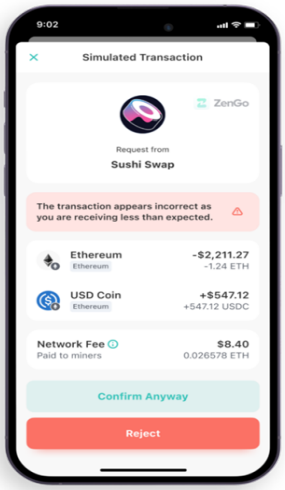
Web wallets offered by Coinbase may also be vulnerable to hacking if appropriate security measures are not taken.

3.2 Vulnerabilities.

3.2.1 I will focus a little bit on the new Coinbase vulnerability.

- Coinbase is a leading cryptocurrency exchange that allows users to store, manage and buy ERC-20 tokens, bitcoins and etherium through its wallet app and ZenGo discovered a Red Pill attack that exploits security vulnerabilities in Coinbase and other cryptocurrency wallet providers.
- Transaction simulation is a common security feature on Web3 platforms that uses sandbox emulation to predict the results of cryptocurrency transactions before they are executed.
- Its main purpose is to prevent fraud and cryptocurrency theft by allowing users to test and preview their transactions before initiating them.

3.2.2 Red Pill Attack.



ZenGo identified a technique called the "red pill attack" which uses transaction simulations to steal cryptocurrency, an attack based on the fact that the malware detects that it is running in a simulation, and this allows it to bypass malware protection and only reveal its malevolence when performing the attack in a real environment.

The ZenGo Wallet report says that some malicious smartwares can be detected when stimulated and exhibit unusual behavior to appear harmless, thereby fooling web3 emulation security.

Analysts explain that attackers can inject "red pills" into malicious contracts to change their simulation behavior and steal money from the victim when they are convinced that the contract is real.

This attack is carried out by filling variables in the smart contract with "safe" data during the simulation and then replacing them with "malicious" data during the real transaction.

These actions will cause the simulation to show the smart contract as secure during the simulation, but will steal the users' cryptocurrency during the actual transaction.

Smart contracts can also be used by attackers for their own purposes, such as stealing sent cryptocurrency or emptying a wallet. Distinguishing between malicious and real requests when signing a contract is difficult, creating a problem for cryptocurrency holders trying to avoid potential dangers.

Here are six cryptocurrency wallet applications that have been found to be vulnerable to "red pill attacks."

ZenGo Wallet, is a Coinbase wallet, Rabby Wallet, Blowfish, PocketUniverse and Fire Extension.

3.2.3 CVE-2022-32969.

Now on to more serious vulnerabilities CVE-2022-32969 .

MetaMask and Phantom warn of a new "*Demonic*" vulnerability that could reveal the cryptocurrency wallet's secret recovery phrase, allowing attackers to steal NFT and the cryptocurrency stored in it. The Demonic vulnerability is tracked as CVE-2022-32969 and involves web browsers saving the contents of input fields without a password to disk as part of their standard "session recovery" system. With Google Chrome and Mozilla Firefox, browsers will cache the data entered into text fields (except for password fields) so that the browser can recover the data after a crash using the "Restore Session" feature.

Because browser wallet extensions such as Metamask, Phantom, and Brave use an input field not labeled as a password field, when a user enters their recovery phrase, it is saved to disk as plain text.

An attacker or malware with access to a computer can easily steal the passphrase and import the wallet into their devices.

An attack would require physically stealing the computer, gaining remote access, or compromising it with a remote access Trojan, which is not uncommon in targeted attacks.However, if the hard drive is encrypted, even if it is stolen,

the attacker cannot access the recovery phrase without having the decryption key.

Users are vulnerable if :

1. The recovery passphrase was imported into the browser extension wallet using a device that no longer belongs to the user or is logically compromised.
2. The hard drive is not encrypted.
3. The user used the "Show Secret Recovery Phrase" checkbox to view the original phrase on the screen during the import.
4. Without the user unlocking their wallet.
5. After rebooting the system.
6. After removing crypto-wallet browser extension.
7. After uninstalling and reinstalling browser.

Metamask fixed the problem in extension version 10.11.3, $xDefi$ fixed it in version 13.3.8, and *Phantom* fixed it in April 2022. Brave has not yet released any statement regarding the vulnerability.

This vulnerability could be quite widespread, as many people use this feature to confirm the entry of the correct word, because often these phrases are long and it's easy to misspell them.

3.2.4 CVE-2018-10812.

In CVE-2018-10812 - Bitpie app to version 3.2.4 for Android and iOS uses open-text storage for initial digital currency keys, allowing local users to steal currency using root access to read /com.biepie/shared_prefs/com.bitpie_preferences.xml (on *Android*) or the plist file in the app data folder.

Bitpie:

It is an integrated blockchain asset service product, and its main functions include: sending and receiving, buying and selling, speeding up transactions, exchanging, and more. It is currently the most popular mobile digital currency wallet in China, which supports multiple cryptocurrencies.

After a brief analysis of the Bitpie app (Android and iOS up to version 3.2.4), researchers found that Bitpie actually stores all the user's initial digital currency keys as plain text. When first launched, bitpie will randomly generate an initial 128-bit entropy and then expand the 128-bit entropy to 132-bit and generate 12 words to help the user remember random bits. Android bither stores the user's initial 128-bit entropy (clutter) as plain text in the file /com.biepie/shared_prefs/com.bitpie_preferences.xml. Even if the user sets a password, the file is still unencrypted.

Where:

```
\< string name="seedPhraseEntropy" \> 0b506e43a5f4a53b8a370984dfb0ec89
   \< /string \>
```

Initial entropy, and according to the initial entropy we can easily reconstruct my 12 mnemonics

Also in the file /com.biepie/shared_prefs/com.bitpie_preferences.xml:

```
\< string name="seed" \> b8409639eba3c15499e66aaa043ac0f512a49adce99be828d405147533c60a
   \< /string \>
```

It is actually a user seedkey that can be used to get the user's root keys of all his cryptocurrencies. The iOS version of Bitpie has some problems, so there is a plist file in the data folder of the app.

And it contains:

```
\< key \> seedKey \< /key \> \
< string \> 889183F3C372B41CAE4025A101FA0022D774z719FEF54F4E91D8C1EA3836710C41A1B434F18
   \ <
```

If an attacker has *root* privileges in an android or ios file. He can simply read these files and steal all the user's coins

3.2.5 CVE-2018-6353 and CVE-2018-1000022.

CVE-2018-6353 .

The *Python* console in *Electrum* before 2.9.4 and from 3.x to 3.0.5 supports arbitrary *Python* code without regard to social engineering attacks, in which the user inserts code they do not understand. Code inserted physically by a direct attacker on hardware left unattended will make it easier for attackers to steal bitcoins with a trap code that is triggered later when the wallet password has been entered.

CVE-2018-1000022 .

Electrum Technologies GmbH Version 3.0.5 of Electrum Bitcoin Wallet contains a "No Authorization" vulnerability in the JSONRPC interface that could lead to bitcoin theft if the user's wallet is not password protected. This attack can be exploited through a victim who must visit a web page with specially crafted javascript, but this vulnerability has been fixed.

- While the electrum daemon is running, someone on another web server virtual host can easily access your wallet through the local RPC port.
- There is currently no security/authentication, which gives attackers access to the RPC port for full access to the wallet.

3.2.6 CVE-2021-41848.

In CVE-2021-41848 - *Luna* mishandles software updates, so that local third-party applications can provide a fake software update file that contains an arbitrary shell script and an arbitrary ARM binary, where both would be executed as *root* user with a *SELinux* domain named *osi*.

To take advantage of this vulnerability, a local third-party application must have write access to an external repository to write a fake update to the expected path. The vulnerable system binary (`/system/bin/osi_bin`) does not perform any authentication of the update file other than verifying that it is encrypted with the AES key (which is hardwired into the vulnerable system binary).

Processes running with the $osiSELinux$ domain can programmatically perform the following actions:

1. Install applications, grant runtime permissions to applications (including permissions with protection levels of dangerous and under development)
2. Gain access to extensive personally identifiable information (PII)
3. Using program permissions, remove applications
4. Set the default launcher application to a malicious launcher application that spoofs other applications
5. Install a network proxy to intercept network traffic, offload kernel modules
6. Set the default keyboard to a keyboard with keyboard logging, notification content checking, text messaging, and more.
7. A fake update can additionally contain an arbitrary ARM binary file, which will be stored locally in internal memory and executed at system startup, to ensure that code is always executed as root user with the $osiSELinux$ domain.

3.3 Safety Precautions .

As you can see, the wallets are not as secure as you might imagine, and some even keep secret phrases in the public domain and an intruder can easily get hold of them.

I can advise you to install only new versions of bitcoin wallet applications, as well as to take personal security measures to keep your data safe (for example, encryption of the dic):

1. one important measure is to choose a wallet where private keys are stored in encrypted form, which provides an additional level of security.
2. .Users should also consider adding additional security measures to their smartphones, such as a fingerprint authenticator, to prevent unauthorized access.
3. strong and unique passwords can also protect against hacking, and many wallet options offer additional security features such as two-factor authentication.
4. It's important not to panic and it's advisable to think through an additional plan to respond to any unusual wallet activity or potential data breach.

5. Another option to increase security is to use a hardware wallet, such wallets are a physical device that stores private keys and requires a PIN to access. Although they are not vulnerable (which we will discuss later), they are nevertheless much more secure)

4 Social Engineering Attacks.

We should not forget about social engineering attacks, now I will talk more about them and give some common examples:

Social engineering in the crypto world is a method of manipulating people into revealing information or doing things that can compromise the security and privacy of cryptocurrency networks, which involves exploiting people's trust, naivety and willingness to believe what they are told.

I will first discuss how social engineering works, and then we will talk about the most common and interesting of its methods.

Social engineering attacks usually consist of two basic steps:

1. First, the attacker researches his target to obtain important data, such as potential entry points and vulnerable security protocols needed to proceed with the attack.

2. Then they work to gain the victim's trust before moving on to actions that violate security rules, including revealing sensitive data or granting access to vital resources.

Social engineering attacks are remarkably simple.

All the attacker has to do is convince an unsuspecting, hurried or gullible person to follow his instructions.

In one of the most famous social engineering attacks, attackers convinced Twitter employees to give them access to confidential processes, later hijacking known accounts such as Joe Biden, Ilon Musk, Bill Gates and Kanye West to trick huge numbers of followers into transferring bitcoin funds directly to the attackers.

These malicious attacks are strikingly simple to execute and follow a similar pattern:

1. An intruder may pose as a senior employee or supervisor.

2. The attacker sends phishing emails en masse.

3. An intruder can gain credibility and then disappear after stealing money.

4. Scareware (scareware, which masquerades as an anti-virus or built-in security system), which scare the potential victim (for example, by saying that he has a virus on his device.

5. An intruder can play on your passion (greed, curiosity).

4.1 Phishing attack variants.

Targeted phishing attacks target specific individuals, organizations or businesses.

- For example, attackers can customize their emails or messages by knowing a person's position within an organization.

- So-called "*vishing attacks*" use voice communications, especially Voice over Internet Protocol (VoIP) solutions, to trick victims into calling and revealing personal information, such as their credit card number or billing address.

- "*Smash attacks*" use SMS or text messages to redirect victims to malicious sites or to trick them into divulging sensitive personal information.

4.1.1 Impersonating an employee.

During a phishing attack, the attacker pretends to be a trustworthy person or organization (for example, he makes a phone call and introduces himself as an employee or owner of a particular wallet) and, under the pretext of blocking the wallet, gets the private key.

Then the attacker tries to trick the victim into revealing confidential information or into giving up their money. Although a phishing attack is often targeted at a specific individual, in most cases the attacker tries to compromise one or more systems which the victim has access to. If a phishing attack on an individual is successful, the consequences can be dire, affecting other users.

Options for phishing attacks include targeted phishing, vishing, and smishing. Targeted phishing attacks target specific individuals, organizations or businesses. For example, attackers may customize their emails or messages to target certain employees of an organization.

4.1.2 Vishing Attacks.

So-called "*vishing attacks*" use voice communications, especially Voice over Internet Protocol (VoIP) solutions, to trick victims into revealing personal information, their card number or payment address during a call.

4.1.3 Smash Attacks.

"*Smash attacks*" use SMS or text messages to redirect victims to malicious sites or trick them into divulging sensitive personal information.

Here's an example:

The March Hare receives a text message from what appears to be his wallet, saying "you need to update your wallet," and inserting a link that the Hare needs to click to "update."

As soon as the silly March Hare clicks the link and fills out the wallet update form, all his data goes to the Queen of Spades.

Likewise, attackers can make a phone call where, under the pretense of hacking your wallet, they ask for the data they need.

A more practical example:

Not too long ago, attackers took advantage of a flaw in the Coinbase SMS account recovery process that gave them a two-factor authentication token in text form. In turn, this vulnerability gave criminals access to thousands of Coinbase accounts and their cryptocurrency.

There is no information yet about the funds lost using this scheme, but Coinbase, has set up a hotline for customers whose accounts have been referenced.

Social engineering attacks targeting large companies often allow access to protected information, leading employees to believe that they have been contacted by a senior employee or executive. As in the case of Coinbase, attackers subsequently use this information to access password-protected user accounts.

Here's one of the victim's testimonials:

"In a letter, Coinbase told affected users: " *While we cannot definitively determine how these third parties accessed this information, these are typically phishing attacks or other social engineering techniques to trick the victim into unknowingly revealing login credentials.*"

Coinbase isn't the only one that has fallen victim to social engineering attacks.

In 2020, a team of attackers ran a similar social engineering campaign on Twitter.

Attackers infiltrated Twitter's server systems, gaining control of 130 verified accounts linked to celebrities, including Jeff Bezos, Kanye West and Barack Obama, as well as prominent companies such as Apple.

The hackers most likely used their newfound followers to swindle hundreds of Twitter users out of more than $100,000 in bitcoins through giveaway scams.

And here's an example of such attacks

4.1.4 A game of trust, and then stealing money.

Scammers find victims by using social networks, dating sites, or messaging apps to gain their trust and introduce them to an investment scheme that will eventually allow attackers to empty victims' cryptocurrency wallets.

These sophisticated scams, where scammers grammatically develop relationships with their victims over a long period of time, have ruined more than a few people and cost people their life savings. As a rule, it is a pyramid scheme, however, such schemes can also work with the theft of the purse. Often, in order to gain the trust of the victim, the attackers share false sad stories.

How does an attack usually take place?

1. The attacker texts the victim on social media wanting to just chat.
2. Later establishes an emotional connection with her.
3. Then, after several months of communication, when the victim does not expect the attack forgives her to either invest or give up her personal information. The unsuspecting victim agrees, and the attacker receives the money and disappears.

4.2 Phishing Exchanges.

At the stage of writing my article, a comrade suggested that I describe another interesting phishing scheme. **This scheme plays on the naivety and curiosity of the victim.**

Here is an example:

1. An attacker creates a site ostensibly a crypto exchange client wallet and sets there a few accounts with a normal balance.
2. The perpetrator offers the victim to exchange funds at an "undervalued rate", he told the victim they "need to quickly get money until the rate is so.
3. the victim transfers money to the purse of the attacker in the pre-specified currency (approx. usdt/bitcoin), and in return receives access to the account at the attacker's site (just she herself can register at the malicious site) .
4. Then the attacker draws a fake transaction from her account to the victim's account.
5. After the victim who has lost part of the funds tries to withdraw money from the attacker's site, but encounters a problem with the need to have a certain amount on the balance, which lacks some part, the victim once again transfers money to the attacker through the top-up form on the site and of course.

6. Naturally, no money does not come, when the victim begins to write to the attacker or possibly in support of his fake site, they either disappear, or send her .

7. The victim is left garrulous and without her money.

4.2.1 Conclusion.

Be more relevant and check the sites where you are going to exchange money, as well as be more wary of "*profitable offers*".

4.3 Using Social Media.

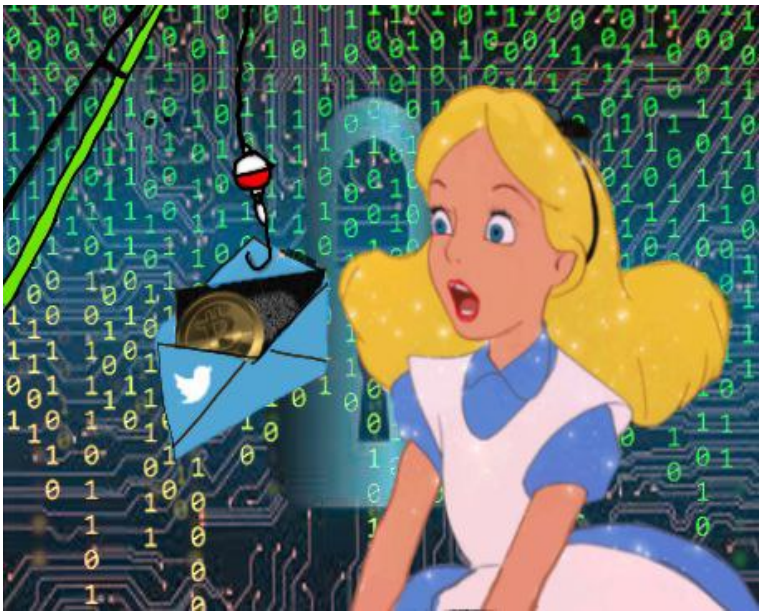
A few common crypto scams on Twitter include:

- Impersonating official and well-known personalities, such as crypto-idea leaders, project managers, developers, and so on.
- Honeypot scams promise impressive rewards for completing basic tasks, but the user must deposit cryptocurrency first.
- Cryptocurrency giveaway scam: you win a big prize on an exchange you've never heard of.
- Fake airdrops for non-existent tokens, phishing links, fake websites and landing pages.

4.3.1 Impersonating official accounts of famous people.

An intruder creates a cryptocurrency account on Twitter or other social networks, then promotes it (recently it was possible to buy an official account icon).

Because of the large number of subscribers and ticks, the victim trusts the fake acanthus on social networks, and then she receives messages with phishing emails in order to get her information.



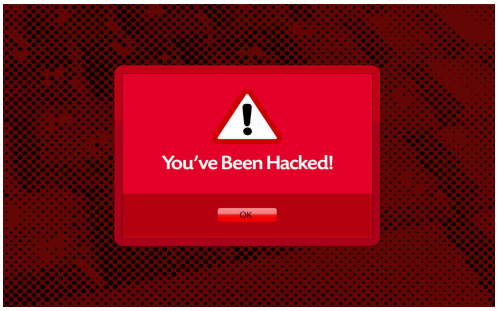
Here's an example:

- Alice receives a fraudulent message in her Twitter inbox.
- Initially, she receives a message in her inbox informing her that she has won a certain amount of cryptocurrency.
- Fraudsters usually use popular and valuable cryptocurrencies such as Bitcoin or Etherium, the message contains a code, Alice is asked to copy it and go to a certain crypto exchange.
- "*Alice simply has to create an account and then activate her code. Once redeemed, the exchange will deposit the specified amount directly into her account.*"
- This cryptocurrency exchange asks you to confirm your email address.
- Given that this is a fraudulent site, the login is needed to convince Alice that the platform is real, but if Alice uses a real email address, then the attackers can save and use it for other attacks (spamming).
- She then enters her bitcoin wallet information, and the status changes to "*Pending.*"
- So far so good - Alice "*just needs to wait a little while for the withdrawal to be confirmed.*"
- This should make her realize that it's all a scam, or suspect something is wrong. Cryptocurrency exchanges usually process transactions instantly,

rather than asking you to wait for administrator confirmation, that's where the scam really happens.

- Alice's bitcoin withdrawal switches to "*Failed*" and the only way to solve this problem is to deposit $0,02$ BTC or $0,3$ ETH.
- "*Depositing the required amount of cryptocurrency will confirm your account and allow the site administrator to confirm your withdrawal.*"
- Poor, naive Alice will transfer money to the intruder, but there will be no withdrawal. Also, instead of simply stealing money, she may enter the data of her real account, which later will be compromised.

4.4 Scareware .



Scareware scares victims into believing it is a serious threat.

For example, you may get a message that your device is infected with a virus.

These are mostly pop-ups in your browser.

Victims are supposed to click a button to either remove the virus or download software that will handle the virus, but this is what leads to the installation of the actual malware.

In most cases, the fraudulent scheme usually ends up with the scammer stealing the users' money and/or information.

Here's what a typical scareware victim does:

- Visits a website and a pop-up suddenly appears (writing that there is a virus on their device), such a message can mimic a genuine security software notification to convince the victim to click on it or perform another action.
- The user either clicks on the link or tries to close the ad, which can sometimes cause the malware to download (usually through fake buttons).
- Pop-ups continue to appear, confirming the relevance of such a virus, this forces the user to act quickly, sometimes deleting the file.

- The user may pay for the associated malware, thinking that this will solve the problem, although the only goal of the scammers was to lure the user out of the money.
- If the user is talked into downloading the malware, there are several ways from here: Scammers can collect sensitive information such as card details, force constant ads on the user, force them to go to unwanted websites or make them download even more malware.

4.5 Attacks that provoke greed and curiosity.-

The least common type of social engineering that relies on human curiosity, luring victims with the offer of a freebie.

Attackers often exploit the victim's greed by promising a quick payout. For example, an attacker might leave an infected flash drive in a public place, hoping that the victim might insert the flash drive out of curiosity and thereby install the malware on their system.

Internet advertisements can trick a victim into promising a quick cash payout in exchange for creating an account with sensitive personal information.

P2P Web sites are also subject to decoy attacks.

The promise of free movie or music downloads can let the user's guard down and refuse to give up information of interest to the attacker.

Victims who have given away information in exchange for the promise of deals, quick ROI, or free cash prizes may later find their accounts empty, but it will be too late.

4.6 Recently there was a funny attack aimed at people's greed.

- The scheme is simple, the user writes to the person, he is introduced as a newcomer to cryptocurrency and says that he can exchange usdt.
- Later he sends the user his secret phrase and private key.
- The user driven by greed goes to the wallet, and sees a certain amount of money there.
- But "here is not the problem" to withdraw it to his main wallet, it is necessary to pay a fee (most often in thorns or etherium or bitcoin).
- Before that, the attacker arranged a smart contact to forward any funds sent to his address.
- This way usdt cannot be withdrawn, but the attacker can get the funds from the commission. The user transfers his money as a commission, but it does not come, in the perplexity of the victim can transfer the commission several times, and in the end just lose his money

Here is another example:

1. The attacker sends a message which reveals the private key or the initial number of the address.

2. At this point an unsuspecting user runs straight to insert a private key or SID into a wallet, usually the one recommended by the scammer, and sees $3,000

3. the victim sees that there are 32 thousand Minerum tokens.(if the victim had checked where we can exchange them, he would have found that HotBit has MNE/ETH pair)

4. Unfortunately, the victim, gripped by greed to get their hands on more than $3,000, neglects these checks.

5. So the first thing she does is make a transaction to withdraw the tokens.

6. But the victim is greeted with a message that the ETH balance to pay the fees is too low, so he will have to send more to be able to make the corresponding transaction.

7. So, the victim transfers just over 0.02 ETH to cover the transfers to HotBit.

8. But, the attackers are quick to transfer any Ethereum revenue transactions to another address.

9. In the end, the victim is left garrulous and without 0.02 ET

4.7 Prevention of crimes related to social engineering.

1. Do not open email attachments from suspicious sources, even if you know the sender and the message does not seem suspicious to you, it is best to contact that person directly to confirm the authenticity of the message.

2. use MFA. One of the most valuable pieces of information that attackers look for is user credentials, using MFA helps ensure that your account is protected in the event that your account is compromised.

3. Follow Computing Services' instructions for downloading DUO two-factor authentication to add another layer of protection for your account.

4. Be wary of tempting offers. If an offer seems good to be true, it is most likely a scam. Use a search engine to read reviews that will help you quickly determine if you are dealing with a legitimate offer or a scammer.

5. Install and update your antivirus, make sure automatic updates are enabled. Check periodically to see if updates are installed, and scan your system daily for possible viruses.

6. Use multi-factor authentication where it is available.
7. Use strong passwords (and a password manager) to protect your accounts.
8. Update your software and operating system, as this can help address security vulnerabilities.
9. Be careful what information you post on social media and other public forums - it can be used for fraudulent purposes.
10. Do not share personal information. - Be very careful how you make friends online.

5 Malware.

5.1 How does it work?

Most often such malware steals information from the PC, browser cookies, keystrokes, etc. (e.g. keylogger).

The malware on a smartphone runs in the background and records every keystroke by the user, which it then sends to the attackers, and the same happens to browser cookies.

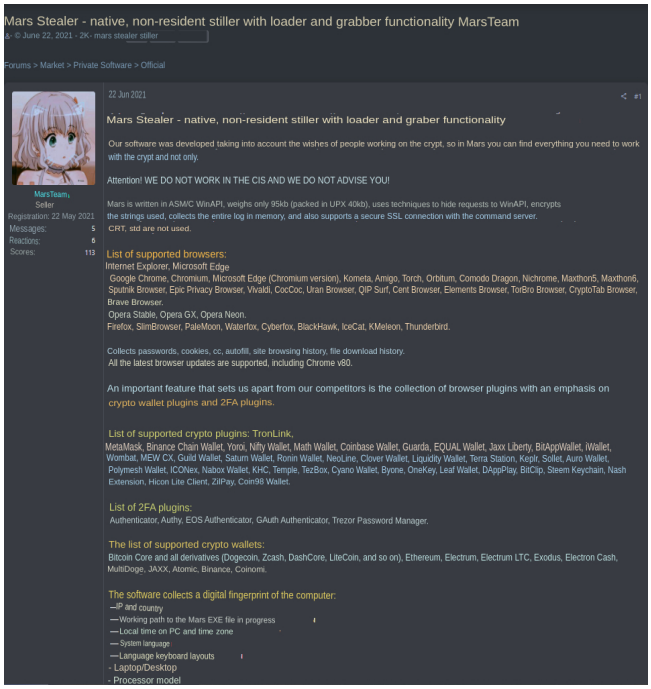
Attackers steal saved passwords, and if you had a saved crypto wallet password, there is a risk that it will be stolen.

There is also a form of malware that overlays a screen that tricks the owner into entering a private key or passphrase on a malicious screen or field within the wallet app.

The malware then sends the information to the attackers, or uses that information directly to take over the wallet and send cryptocurrency funds to the attackers' accounts.

As you can imagine, security has always been an issue for browser-based cryptocurrency wallets that store cryptocurrencies such as Bitcoin and Etherium, but the new malware further exacerbates the security issue by targeting wallets that work as browser extensions (some examples are MetaMask and Coinbase Wallet).

5.1.1 Mars Stealer.



One of the most common malware threats, called Mars Stealer, and is an update of an older 2019 Trojan that targets more than 40 browser-based cryptocurrency wallets with a looting capability designed to steal a user's private keys.

This new wallet-stealing malware spreads through many different online channels, including file sharing sites, torrent clients and other unverified download sites.

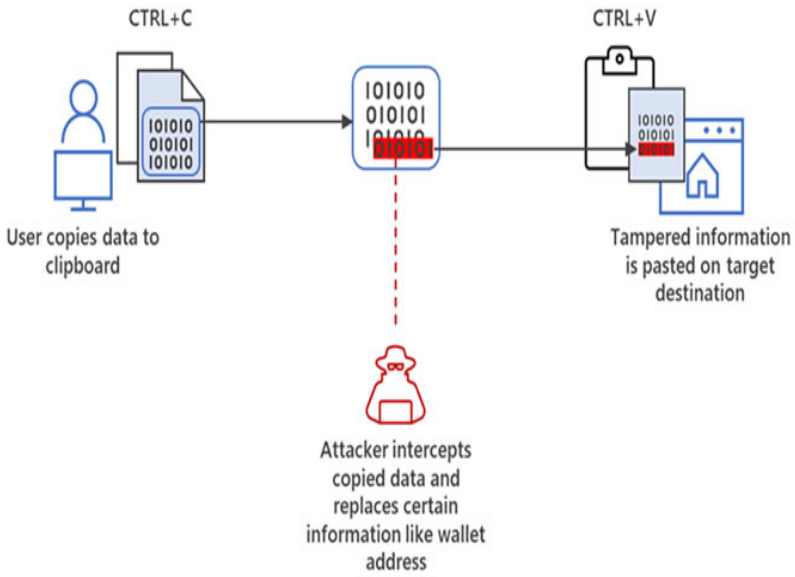
The stealer works by targeting the user's device file where the private keys are stored, stealing information and removing any presence of the theft.

Mars Stealer can be purchased on the darknet for as little as $140, attracting potential attackers.

Currently, the malware only targets hot wallet credentials from Chrome, Firefox and Opera browsers, which at first glance don't seem vulnerable to

extension-related cyberattacks, but they could still be a target for site credential hacks.

5.1.2 Cryware.



A new threat known as Cryware, a form of malware designed to collect and extract data from hot wallets or cryptocurrency wallets that are stored on common user devices and help perform cryptocurrency transactions, has recently emerged.

Cryware and other Information Theft Trojans represent a shift in attacks targeting cryptocurrency by helping hackers access hot wallet data and quickly transfer cryptocurrency financial assets into their own wallets, and because these are blockchain transactions, the theft is irreversible.

Cryptocurrency wallet malware works slightly differently for Android or Apple phones:

If victims use **Android** phones, those victims are usually new cryptocurrency users who don't yet have a legitimate wallet app on their phone.

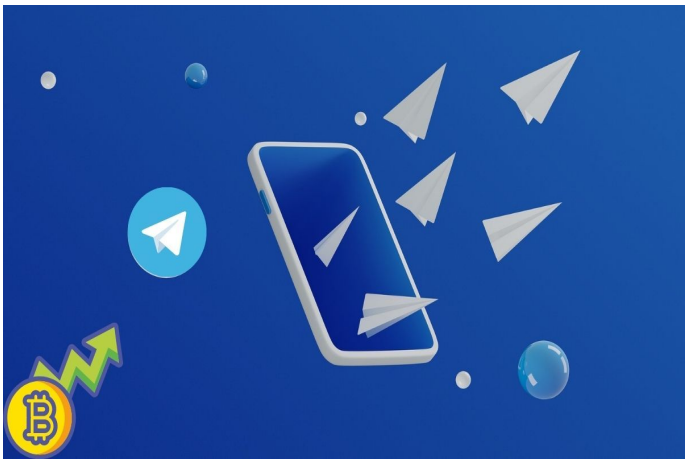
Since the Android security protocol does not allow malware to overwrite an existing app, users are tricked into accessing a fake crypto site (presumably from Google Play), but it is actually downloaded from the fake site's web server.

Similarly, **Apple's** protocol prevents them from downloading the malicious app from the Apple App Store, victims are redirected to a third-party website that downloads the malware. Hackers

send alerts and notifications to convince the user to bypass the iPhone's built-in protection and install the malicious app without their knowledge.

5.2 Here are examples of such viruses.

5.2.1 Echelon



The Echelon malware sample detected by the researchers was found in a .rar file.

It included three different files:

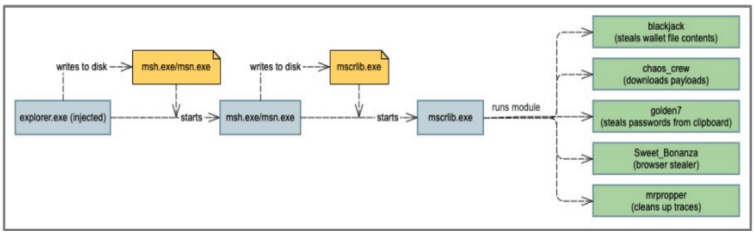
1. 123.txt, a text document containing the password
2. dotNetZip.dll - a small non-malicious class library for ZIP files management
3. Present.exe- a malicious executable file from Echelon malware.

The detected executable was obfuscated with ConfuserEx.

- Once launched, the malware steals credentials and takes screenshots.
- The Echelon malware sample was posted on a Telegram channel related to cryptography, no exact number of victims is known yet, but it is clear that by posting its virus on a specialized Telegram channel.

The attacker was targeting people familiar with cryptocurrency here is a satay about it

5.2.2 BHUNT



A cryptocurrency theft malware called BHUNT is another cryptostealer added to the big pile of malware targeting digital currencies, but it deserves special attention because of its stealth.

- The main component of BHUNT is "*mscrlib.exe*," it extracts additional modules that run in an infected system to perform various malicious actions.
- BHUNT is injected into *explorer.exe* and is likely delivered to the compromised system via the download of KMSpico, a popular utility for illegally activating Microsoft products.
- To avoid detection and triggering security warnings, BHUNT is packaged and heavily encrypted with *Themida* and $VMProtect$, two virtual machine packers that prevent backward design and analysis by researchers.

The attackers signed the malware executable with a digital signature stolen from $Piriform$, the creators of *CCleaner*.

- Each of its modules is designed for a specific purpose, from stealing cryptocurrency wallets to stealing passwords.
- Using a modular approach, attackers can customize BHUNT for different campaigns or easily add new features.
- Current modules included in the BHUNT mscrlib.exe executable

Once an attacker gains access to the wallet's initial state or configuration file, they can use it to import the wallet into their devices and steal the cryptocurrency it contains.

While BHUNT's focus is clearly on finance, its ability to steal information may allow its operators to collect much more than just crypto wallet data.

Articles about it:

- `https://www.welivesecurity.com/2022/03/24/crypto-malware-patched-wallets-targeting-andr`
- `https://www.bleepingcomputer.com/news/security/new-bhunt-malware-targets-your-crypto-wa`

5.3 Malicious wallet application.

ESET research identified more than 40 copycat websites for popular cryptocurrency wallets.

These websites target only mobile users and offer them downloads of malicious wallet apps. They were able to trace the distribution vector of these Trojanized cryptocurrency wallets back to May 2021 based on the domain registration that was provided for these malicious applications in the wild, as well as the creation of several Telegram groups, they began searching for affiliate partners.

5.3.1 The malicious app behaves differently depending on the operating system in which it is installed.

On **Android**, the malware targets new cryptocurrency users whose devices do not yet have a legitimate wallet app installed.

Trojan wallets have the same package name as legitimate apps, but they are signed using a different certificate, which means that if a legitimate wallet is already installed on an Android smartphone and the malicious app cannot overwrite it because the key used to sign the fake app is different from the legitimate app. This is a standard security model for Android apps, whereby non-original versions of the app cannot replace the original.

On **iOS**, however, a victim can have both versions installed - the legitimate one from the AppStore and the malicious one from the website - because they don't use the same package identifier.

For both platforms, the downloaded apps behave like full-fledged wallets - victims and see no difference. This is possible because attackers have taken legitimate wallet apps and repackaged them with additional malware.

5.3.2 How does it work?

Repackaging of these legitimate wallet apps had to be done manually, without the use of any automated tools.

- Because of this, the attackers needed to first perform an in-depth analysis of the wallet apps for both platforms, and then find the exact places in the code where $seed-phrase$ is either generated or imported by the user.
- In these places, the attackers inserted malicious code responsible for getting the $seed-phrase$ and extracting it to the attackers' server. If the attackers have the $seed-phrase$, they can manipulate the contents of the wallet as if it were their own.
- Some malicious applications send the victim's secret source phrases to the attackers' server over the unsecured HTTP protocol, without any additional encryption. Because of the insecure protocol, other attackers on

the same network can eavesdrop on network communication and steal the victims' initial or recovery phrases to gain access to their funds, this attack scenario is known as a *mitm* attack.

- The malicious code was patched into the *classs.dex* of the malicious Android wallet, meaning that a new class was inserted, including calls to its methods found in certain places in the wallet code where it handles the $seed - phrase$.
- This class was responsible for sending the $seed - phrase$ to the attackers' server; the server names were always hard-coded, so the malicious app could not update them if the servers were disabled.
- In an iOS app, the attacker injected a malicious dynamic library, dylib, into a legitimate file; such a library becomes part of the app and is executed during startup.
- The *libDevBitpieProDylib.dylib* dynamic library contains malicious code responsible for extracting the victim's $seed - phrase$.
- The malicious code is not always present in compiled form; some of the wallets are web apps, and their mobile apps contain all web components, such as $HTML$, images, and scripts, in assets within the app.
- In these cases, attackers can instead insert malicious code into *JavaScript* and this method does not require modifying the executable.

5.4 Here are a couple of examples from github.

5.4.1 Leaf-Stealer

Account and bitcoin wallet thief (captures other logins from sites including steam, paypal, Sellix) after launch, when it infiltrates victim's computer, gets always new information after reboot (undetectable by antivirus).

What it steals:

- Cookies;
- Passwords;
- 2FA codes; -
- Computer hostname;
- Metamask;
- Desktop files (.exe, .png, .rar, .bat, .txt).

5.4.2 Invicta-Stealer

And here is the C++ stealer, which is actively improving and stealing:

- Information obtained from all profiles, chrome-based browsers (most commonly used) and Firefox.
- Attackers collect: bank card details, autocomplete, history, all extensions that include over 80 cryptocurrencies, various authenticators and password managers, local storage, downloads and more (essentially all information is collected).
- All discord tokens are extracted from regular client,discordcanary,ptbdiscord and local browser storage.
- Wallet information is collected from 25 wallets, new ones are actively being added. There are real world scripts and advanced filters that will extract sensitive information related to cryptocurrency wallets, bank accounts, passwords, private keys, etc.
- The thief gets recently opened.txt files, recursively crawls your computer to find sensitive information, steals github and VisualStudio code repositories, gets .txt files from your desktop, documents, etc.

How to use:

1. Download a ZIP file of the builder.
2. Run Builder.exe.
3. Enter the Discord webhook or URL of your HTTP server in the box.
4. Click build.
5. The patched styler will be available in the folder $out/InvictaStealer.exe$.

5.4.3 Bbystealer

A new, advanced tool for capturing and stealing tokens with a password, even when they change can get the following information:

- Functions.
- $QRCode$ Registrar.
- Steals username and id, token, password, email, cookies.
- Performs instant logout. Disables $QR - code$.
- User insertion. Automatic *cookie* logger.
- Anti-deletion and anti-spam protection.

5.4.4 Klimt

Klimt is designed to steal credentials, analyze system information, and assess security.

It has a number of capabilities to compromise different types of systems and applications, including Discord, various cryptocurrency wallets, web browsers, Roblox, and many others.

Klimt also includes a persistent back shell that allows users to execute commands from a remote server, as well as an interactive GUI builder.

Features (this tool has a professional version, and some features are made just for it:)

- Stealing credentials.
- Program Injection.

To use Klimt Builder, follow these steps:

1. In the Klimt directory, run builder or, open a new builder.exe file to open the Klimt Stealer Builder.
2. Wait a few seconds, and when the linker GUI opens, customize it to your liking and click Compile Stealer.
3. If the build is successful, the executable should be in the build/agent.exe folder and it can now use this stub at will.

6 MiTM attacks on crypto-applications.

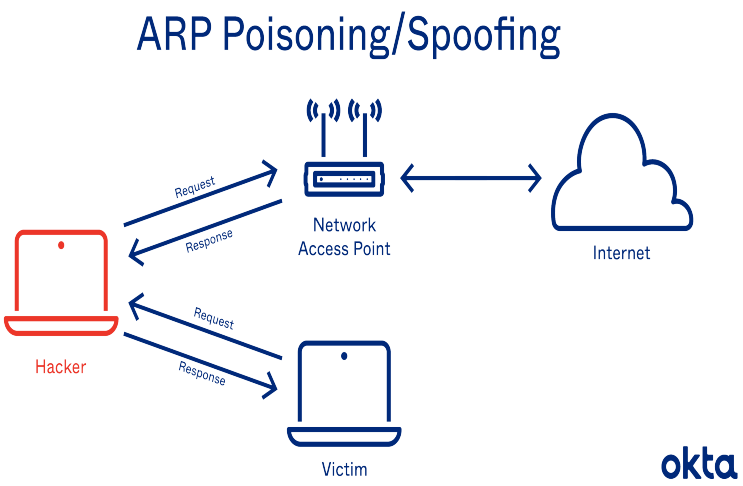
6.1 First I will tell you what MiTM attacks are in general.

Imagine Alice and the rabbit, the rabbit gives Alice data (to make it visually clear, imagine the rabbit throws Alice a watch), between them stands the Cheshire cat, who tries to intercept the watch, and it does not matter who has it, Alice or the rabbit. During the game, the Cheshire Cat can pass black roses (malicious links or other information to both participants) to Alice or the rabbit.

In a mitm attack, the middle participant manipulates data unknown to either of the two legitimate participants, acting by any means to obtain confidential information or cause damage.

6.2 Before we talk about blockchain attacks, I will name the most common types of such attacks.

6.2.1 ARP spoofing.



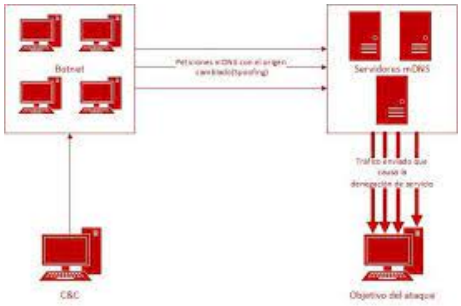
ARP is an address resolution protocol and is used to resolve IP addresses to physical MAC addresses on a LAN.

When a host needs to contact a host with a given IP address, it accesses the ARP cache to convert the IP address into a MAC address. If the address is unknown, the MAC address of the device with the IP address is queried.

An attacker wishing to impersonate another host can respond to requests using its own MAC address, with some precisely placed packets the attacker can listen to private traffic between the two hosts.

Valuable information, such as session token exchanges, can be extracted from the traffic, giving full access to application accounts that the attacker should not have access to.

6.2.2 MDNS spoofing.

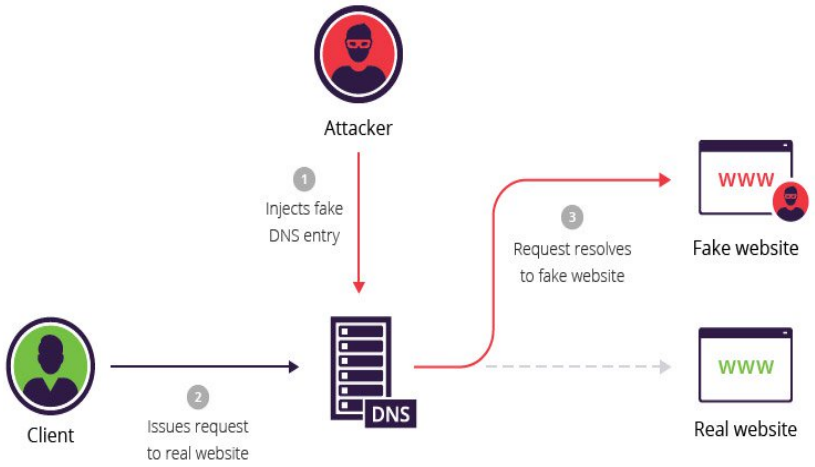


MDNS is similar to DNS, but executed on the LAN using broadcasting such as ARP, this makes it an ideal target for spoofing attacks. The local name resolution system is supposed to make setting up network devices as easy as possible. Users don't need to know exactly what addresses their devices should associate with, they let the system decide that for them. Devices such as televisions, printers, etc. use this protocol because they are usually on trusted networks.

When an application needs to know the address of a particular device, such as tv.local, an attacker can easily respond to this request with fake data, instructing it to allow use of the address it controls.

Because the devices store a local address cache, the victim will now assume the attacker's device is trusted for some time.

6.2.3 DNS spoofing .

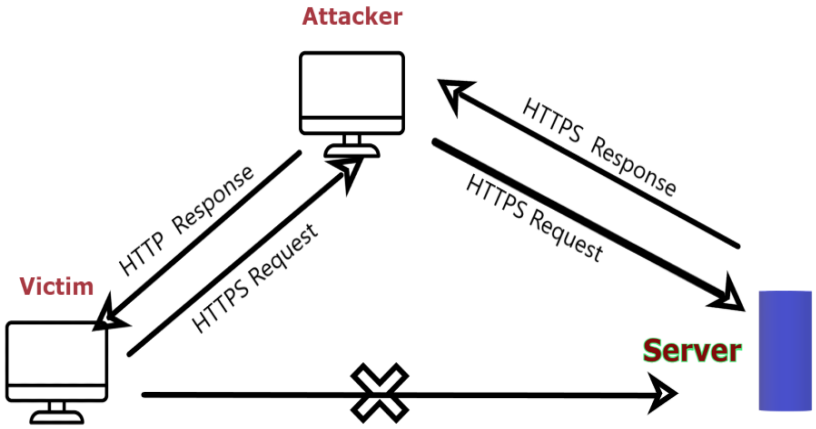


Just as ARP converts IP addresses into MAC addresses on a LAN, DNS converts domain names into IP addresses.

In a DNS spoofing attack, an attacker attempts to inject corrupted DNS cache information into a host by attempting to access another host using its domain name, such as rabbit.alise.tea.

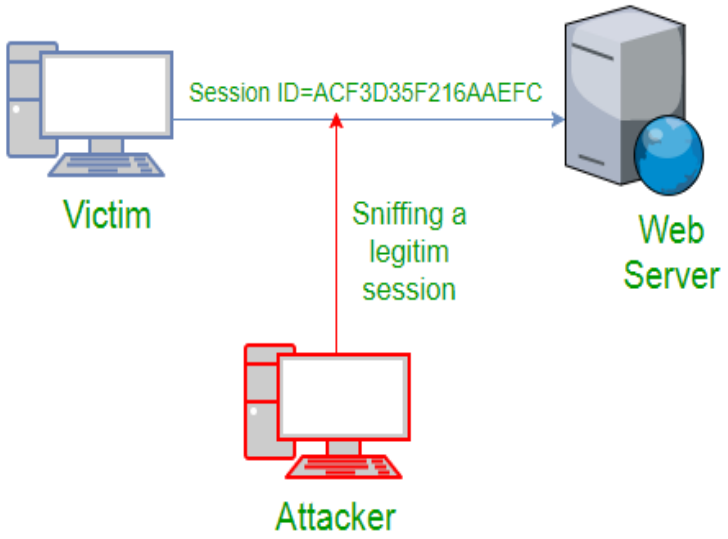
This causes the victim to send sensitive information to the malicious host, believing that it is sending the information to a trusted source. It can be much easier for an attacker who has already spoofed an IP address to spoof DNS by simply converting the DNS server address to that of the attacker.

6.2.4 Deleting SSL.



Since the use of HTTPS is a common defense against ARP or DNS spoofing, attackers use SSL separation to intercept packets and modify their HTTPS-based address requests to go to the equivalent HTTP endpoint, forcing the host to send requests to the server in unencrypted form. Confidential information can be transmitted as plain text.

6.2.5 Session hijacking.



Most web applications use a login mechanism that generates a temporary session token for use in future requests so that the user is not required to enter a password on every page.

An attacker can listen to sensitive traffic to identify the session token for the user and use it to send requests on behalf of the user. The attacker doesn't need to forge anything if he has a session token.

6.3 Now back to our cryptocurrency.

As we remember from the introduction to *Bitcoin*, the user uses string hashes from public keys, which look like random strings, to receive payments.

Unfortunately, there is no authority to verify the user's identity, and usually the user cannot prove that the address is related to his real identity. Technically, a victim can get a fake address and pay coins to that fake address.

Many organizations and individuals post their bitcoin addresses on web pages linked to info-rum posts, blogs, and social networks in order to receive bitcoins.

After the payer transfers the coins to the recipient's address, the recipient redeems the transaction with his private key via the p2p network protocol.

6.3.1 Security Problem

Consider how a MitM attack is susceptible to Bitcoin address falsification over the web using HTTP/HTTPS.

The main problem with the MitM security vulnerability is that most Bitcoin addresses accidentally posted on forums, blogs, merchant websites, social networking sites, and are not properly secured.

6.3.2 Example of an attack.

Let's analyze a Bitcoin transaction between two parties in a less secure channel, where the payer (let's call him Alice), sends an amount of coins to the recipient (Rabbit), and Rabbit redeems the coins.

The conditions for the attack are:

1. The victim publishes his address on his own site without HTTPS protection or without a certificate from a trusted authority.

2. The victim publishes their address on a website or forum post without HTTPS protection or without a certificate from a trusted certificate authority.

The victim publishes his/her address on an HTTPS protected site or forum.

We consider an attacker acting in the middle between Alice and Rabbit during a transaction.

Alice initiates the payment, using Rabbit's Bitcoin address to send the payment. We assume that the communication protocol is vulnerable, so an attacker (the Cheshire Cat) can get an advantage between Alice and Rabbit without their knowledge and spoof the address.

Who can attack you?(Except the Cheshire Cat☺)

> An attacker can be a web service provider, an Internet Service Provider (ISP) or a malicious Wi-Fi access point. The attacker's main task is to determine the right address from its stored content or blockchain.

Consider two categories of attackers:

1. An attacker close to the recipient, such that when the recipient publishes its address through the attacker, the attacker will change the target address to that of the recipient.

2. The attacker gets close to the payer, the payer pokes the attacker for the payee's address, and the attacker easily modifies the address with his own posting.

How can an attacker identify high-value addresses?

> Initially, an attacker can easily filter out potential Bitcoin addresses from web content and then use its internal checksum to verify that it is a Bitcoin ad.
>
> The basic calculations for the attacker are REGEX filtering and SHA-256 double checksum generation.

After receiving the address, the attacker can check the Bitcoin blockchain to make sure it is a persistent address with a large number of coins being received.

The attacker can then replace the address with his own.ь

Ironically, the attacker can use the disposable address to maintain privacy if the certificate is invalid, since his security guarantees depend on the validity of the certificate.

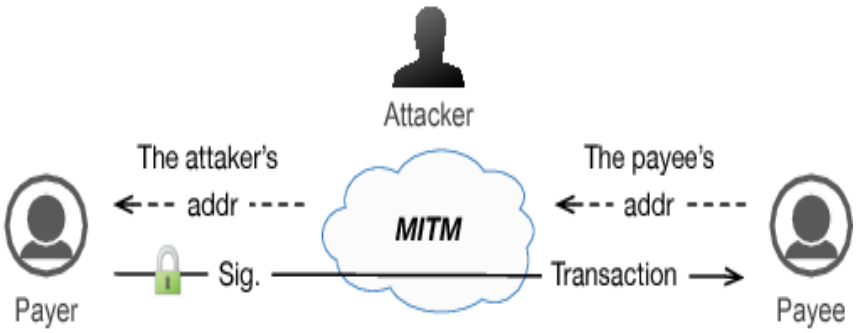
It is very easy for an attacker to generate multiple addresses using some hierarchical address generation methods, in this case MitM is very powerful because by the nature of HTTP connection data can be transmitted as plain text.

An attacker can easily spoof Bitcoin addresses randomly placed on web pages using HTTP.

The attacker can then simply modify the content of the web page containing the Bitcoin addresses and replace them with addresses under his control, HTTPS can suffer the same actions.

Here's a githab thread with tools for such attacks

6.4 Attack on buy/sell transactions.



Not quite a MITM attack, but also a very interesting topic.

6.4.1 How does an attack occur?

In traditional centralized systems, all transactions are verified based on information from a central controlling authority before being executed.

In decentralized systems (cryptocurrencies), there is always the technical possibility of Double-spending. This means that a user can

make multiple payments transferring the same asset almost simultaneously.

Information about them will not instantly get to the next block, as a consequence, the recipient will be sure that he has made a valid transaction.

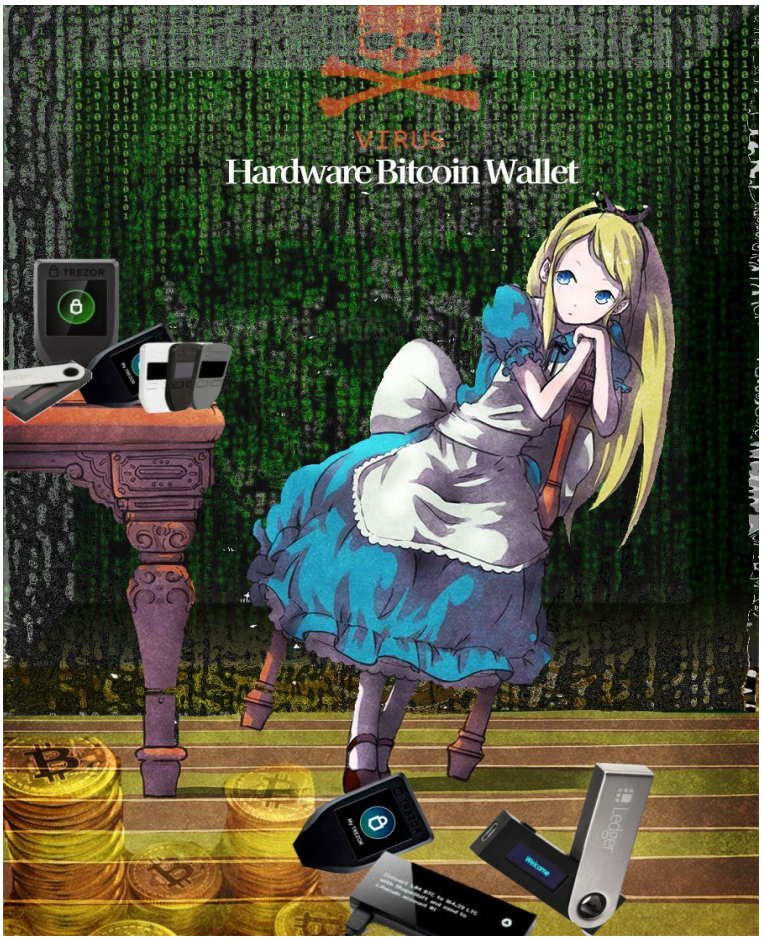
Only after one of the transactions (not necessarily the first in time) is made will the other transactions with the same asset no longer be valid.

For example, after transferring money (cryptocurrency), a malicious buyer expects to receive the product he paid for in return.

But after receiving the product, he can generate a new conflicting transaction that sends the same money to himself.

The second transaction, if accepted by a valid transaction, would leave the seller without both the money and the product.

7 Hacking hardware wallets.



7.1 Cracking PIN code on paper.

If we are talking about bitcoin wallet hacking, I think it is very reasonable to talk about hardware wallet hacking, but first I will tell a funny hack, when a person wrote down the PIN code on paper.

7.1.1 How the attack happens

A paper wallet cannot be hacked due to the fact that it is just a record on paper, nevertheless hackers got to this cryptocurrency repository as well.

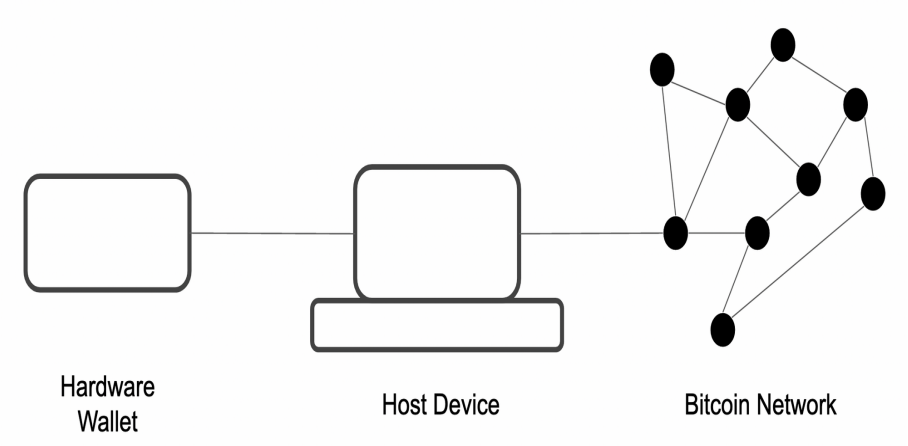
The thing is that many users, after writing down the address and the secret key on paper, photograph them and store them in their smartphone's gallery just in case, and sometimes intentionally or accidentally sync the photo with the cloud storage or with iTunes on their computer.

It should be noted that pictures of secret wallet IDs are stored in the computer along with other pictures, such as pictures of bank cards and their PINs.

Trojan viruses either contain a built-in mechanism for analyzing images or simply select from all the photos the images they are looking for according to basic principles and send them to the attackers' server.

- A simple criterion for selecting photos with secret wallet codes is the prevalence of a white background in the photo.
- This criterion is enough to filter the images on a user's phone so that the virus remains "lightweight" and does not require detailed analysis of the images, which is performed on the attacker's server when the images are obtained using appropriate software.
- Having obtained a snapshot of cold wallet codes, the attacker easily transfers the user's money to his account.

7.2 Why do I need a hardware wallet?



Now let's break down why someone needs a hardware cryptocurrency wallet?

There are many ways to store wallets, each of which has its pros and cons.

The most convenient way is to store sido either on your computer, your smartphone, or, even more convenient, online.

- However, as we saw above, malware attacks on cryptocurrency wallets are not at all uncommon.
- As for online wallet services, they can be hacked and even stolen.
- On top of that, other problems with wallets, including phishing, counterfeiting of payment information, and losing wallets due to hardware failure have become so frequent.

At some point, people solved the problem by creating hardware cryptocurrency wallets (specialized devices designed to safely and securely store cryptographic keys).

- Cryptocurrencies usually organize the possession of funds using public and private key cryptography (or derivatives thereof). Cryptocurrency wallet software manages these private keys and transmits signed transactions to the corresponding blockchain.

Detachable ded- able hardware tokens promise to be a reliable single-purpose solution for storing private keys in a secure vault against unauthorized access or other theft.

While a specialized token still needs a general-purpose PC to conduct a transaction, it is assumed that a compromised computer can no longer issue private keys or modify transactions, but this trust may be unwarranted.

Many hardware wallets of our generation do not operate within a fault-tolerant, mutually verifiable architecture.

Instead, as our research has shown, they simply transfer trust to external hardware, which actually only results in a new single point of failure.

Consequently, they change the security risk profile associated with the PC to the risk profile associated with the hardware token.

Consequently, they miss the opportunity to limit the risks to the intersection of all the interpenetrating components of the PC, the wallet software, and the wallet hardware token involved.

7.2.1 How do hardware cryptocurrency wallets work?

- The basic idea of a hardware cryptocurrency wallet is to store cryptographic information in such a way that it never leaves the device.

- All crypto-signing is done inside the wallet, not on the computer it is connected to, so even if your computer is compromised, attackers will not be able to steal your keys.
- In addition, it would be nice to have some access protection measures - such as locking the device with a PIN.
- It would also be very useful for a hardware wallet user to be able to verify the actual transaction on the device with the ability to either confirm or deny it.

All of these considerations determine the most appropriate action layout, typically a hardware cryptocurrency wallet is a relatively small USB key, it has a display and a few buttons that are used to enter the PIN and confirm the transaction.

However, the inner workings of such devices can vary.

Two leading manufacturers of hardware wallets, Trezor and Ledger, present two different approaches to the hardware approach.

You can hack a hardware wallet in several ways, but in all cases the goal of the attack is to gain access to sensitive data such as your PIN or private keys.

Your secret PIN is a barrier to protecting your data, but that PIN is as secure as the device.

7.3 How can I attack a hardware wallet?

7.3.1 Physical attack: power failure.

The idea behind a power failure is to flood the power to the equipment's circuit board for a short period of time to put the device in a state similar to exhaustion.

With the power surge, the device's circuitry becomes entangled and leaves sensitive information open to an intruder. The microcontroller chip is the gateway to the private keys, and breaking it reveals the secrets (data) inside.

A power failure attack does this by using high voltage current bursts on the component to make it vulnerable, allowing access to the raw data.

From there, it's relatively easy to recover the device's PIN and access the underlying personal data on the chip.

- In short, a power glitch attack is a brute-force attack that requires an attacker to directly access your wallet.
- The principle of a third-party channel attack is to observe the behavior of the hardware wallet during a transaction.

7.3.2 Through radiated information: side-channel attacks.

The principle of a side-channel attack is to observe the behavior of the hardware wallet during the execution of a transaction.

We can compare this attack to a burglar who uses a medical stethoscope to listen to a locked safe and extract information by listening for changes while fiddling with the lock.

To perform a side-channel attack, you use an oscilloscope to monitor the power consumption of the device while it is running.

By listening to the noise of the device and fiddling with random PINs, you can observe how the consumption behaves and how each code changes behavior.

Different PIN values leave a different trace, making it easy to determine which codes might be working. Studying the dynamics of energy consumption with each PIN code attempt creates a database.

This information is used in conjunction with a script that guesses the PINs one by one and is used to crack the code.

The sidelink attack "listens" to the information emitted by your device to figure out its PIN.

Once this is done, the attacker can use your wallet as his own.

7.3.3 Obtaining secrets with software: Hacking hardware using a software attack.

The principle of attacking a hardware security module (known as HSM) is to recover the software underlying its operations to understand how it works.

A software attack means getting to know the technology better than the developers know it and fixing the vulnerabilities in it. It's a process of research and exploitation.

The first step:

Connect the hardware module to your computer, and from there you interact with it to restore its core software.

This is done by running a script that digs into the code of the device to find the software in binary form.

But we can't understand the binary code, so to get the information to a point where it can be understood requires a little reverse engineering to turn the code into something that a human can interpret.

Using this redacted software information, the goal is to try to find a point of vulnerability that can be exploited, allowing an attacker to gain control of the software and get data from it.

Here's an example of an attack:

Researchers studied $LedgerNanoS$ firmware and found that the wallet could be re-flashed with a compromised version by writing a certain value to the right memory address.

The *Ledger* developers protected this address from writing, but it turned out that the microcontroller allows memory addressing changes - that is, the cherished memory cell can be assigned another address, which is not included in the list of blocked addresses.

The researchers took advantage of this hardware feature and loaded modified firmware into the $NanoS$.

For demonstration purposes, this modification contained the game "$Snake$" - but instead it could have been, for example, a malicious module that substitutes the wallet address in all outgoing transactions.

7.3.4 Another way to compromise the hardware wallet is based on hardware implants.

Josh Datko was able to connect an inexpensive implant to the $LedgerNanoS$ that presses an acknowledge button when a malicious command is received over the radio.

Apparently, any hardware wallet can be compromised this way - the researcher chose $LedgerNanoS$ as the victim of his experiment simply because it is the most compact of cryptocurrencies, so it was most difficult to embed the implant in it.

Another device from the same manufacturer, $LedgerBlue$, turned out to be vulnerable to attacks through third-party channels.

$LedgerBlue$ is a hardware wallet with a large touchscreen and a high-capacity battery. It turned out that, due to an unfortunate PCB design, it emits quite discernible radio signals when the user enters the $PIN - code$.

The researchers recorded these signals and pitted them against a machine-learning algorithm, which learned to recognize them with 90% accuracy

One of the most popular hardware wallets today is the $Trezor$ wallet, produced by the Czech company $SatoshiLabs$, which can be used to sign transactions that are confirmed through a small display.

- In other words, in order to make (confirm) a payment, it is necessary to enter confirmation data on this device.
- The wallet works with different computer programs/clients: $MultiBit$,$Mycelium$,$Electrum$
- Moreover, the device can be used to store altcoins.

To start working with the wallet the user needs to connect it to the computer and go to the website of the company $myTrezor.com$ in the browser to follow further instructions.

From the site you download and install the plug-in program for your browser. Wallet is well protected from attacks by keyloggers. In Tezor wallet cryptocurrency is protected from hacker attacks.

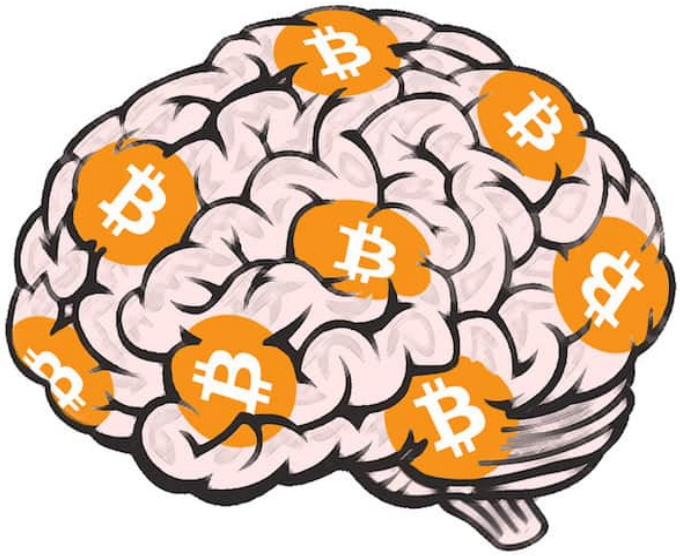
A random code is generated on the wallet screen, and the browser prompts you to enter a PIN code based on the order of the digits of the code on the screen of the device.

Since the order of the digits is generated differently each time, it is necessary to refer to the hardware wallet before each PIN entry.

Another wallet manufacturer, *Keepkey*, has the same problem.

Here is also a list of hardware wallet hacks as examples

8 Hacking brainwallets .



8.1 What are wallets?

In the cryptocurrency Bitcoin, users can deterministically derive private keys used to transfer money from the password.

Such "brainwallets wallets" are attractive because they save users from having to store their private keys on untrusted computers.

Unfortunately, they also allow attackers to conduct unlimited offline password guessing.

Since private keys are not permanently stored on the devices, the "brain" wallets cannot be removed by malware, but there is a downside:

- Anyone who guesses a user's password can steal their funds.
- Worse, attackers can perform unsupervised (offline) guessing to test candidate passwords.
- Attackers who guess a password can quickly verify that it matches any user's wallet by scanning the use of a derived public key in the Bitcoin blockchain, the public ledger of all transactions.

- Moreover, many passwords can be tested there e.g. in one study they tested 300 billion passwords.

8.2 Testing attacks.

8.2.1 Tools.

Attack testing can be done with the help of the project:
Brainflaye
What is Brainflayer :

It is a *brainwallets* wallet hacking tool with proof of concept that uses $libsecp256k1$ to generate a public key.

It was originally released as part of my $DEFCON$ report on brainwallets hacking.

The name is a reference to Mind Flayers, a race of monsters from the $DungeonsDragons$ RPG. They eat brains, psionically enslave people, and look like Lovecraftian horrors.

Features:

- The current version runs more than four times faster than the DEFCON version, and it adds many features.
- It (mostly) does one thing: hunts for tasty brainpurses.
- The main function it doesn't have is generating possible passwords/password phrases.
- There are many other great tools that do this, and Brain Flyer will be happy if you direct their output to it.

8.2.2 Self-Testing.

- The researchers observed using the Bitcoin Brain wallet as the private key we use $SHA256 - hash$ passwords.
- Then they generate the corresponding public key using several accelerations of the $secp256k111$ curve library.
- They loaded the Bitcoin blockchain using Bitcoin core12 software and extracted all unique Bitcoin announcements using the $znort98713$ block analyzer.
- They then added all addresses to the *bloom* filter for a quick search and to a sorted list to detect false positives.
- They compared all addresses generated from candidate passwords with the bloom filter and confirm positive results on the sorted list.

- Once they found all the used brain wallet addresses, they supplemented this information by querying all the brain wallet addresses to the *blockchain.info* API to get the exact timestamps of all the transactions.
- Transactions with brainwallets as recipients are incoming payments, and transactions with brain wallets as sources are outgoing payments.

Having studied all transactions on the blockchain through 8/2015. Researchers reported their prevalence, leakage, and reliability of passwords.

8.3 How common are brainwallets wallets?

Researched, found 884 different basic wallets using 845 different passwords.

The small difference is due to the small number of cases where compressed and uncompressed wallets were used for the same password. A total of 1806 BTC (about $\$103K15$ was deposited into these wallets.)

So as you can see this threat is not small, so be vigilant.

9 Brute Force Attacks.

9.1 Now let's talk about brute force attacks.

As we remember a Bitcoin wallet can be thought of as a set of addresses, which are unique identifiers assigned to the possession of certain funds. A person's

possession of an address corresponds to their knowledge of a secret (also known as a private) key, which is used to create the corresponding address.

Note that one person can have an unlimited number of addresses.

Moreover, for security purposes, it is recommended that a new address be generated after each withdrawal from an existing address.

- A compressed version of the public key, which is a $268 - bit$ string
- After that, an irreversible hashing operation is performed, first using $SHA-256$, and then using $RIPEMD-160$ cryptographic hash functions.
- The result is a $160-bit$ string, which is the essence of the Bitcoin address.
- In practice, this hash is concatenated with 4 bytes of the checksum (obtained by doubling SHA-256) and an extra version byte,
- And then the $25 - byte$ (200-bit) result is presented in $Base58Check$ encoding to produce the final Bitcoin address.

Note that the transition between the $RIPEMD - 160$ hash and the $200 - bit$ address can easily be undone.

Due to the intrinsic properties of cryptographic hash functions, which make them work as pseudorandom functions, the effective size of the address space is given by:

$$Naddr'2160 \approx 1.46 \times 1048$$

Which is much smaller than the size of the space of possible Nsec keys.

The security of the cryptographic hash functions used should make it computationally infeasible to find any valid public key yielding a particular $160-bit$ hash.

In the following, we consider only $Pay - to - PublicKeyHash(P2PKH)$ transactions.

- The idea behind this type of transaction is simple, if a user wants to transfer some funds to a certain address, the victim publishes a special form transaction with an output containing that address.
- In order to redeem the funds, the owner of the address must create another transaction with an input containing a public key that hashes this address and a signature that is created using the secret key and the corresponding public key.
- This signature serves as proof to the system that the author of the transaction is indeed the owner of the address.

9.2 Small Group Attack.

The attack under consideration is based on an exhaustive search strategy for a signature forgery attack, which is conducted by an attacker in order to steal bitcoins from honest users.

1. The attacker makes an AddrList of all bitcoin addresses that have some means using the current set of unspent transaction outputs ($UTXO$) corresponding to the current state of the Bitcoin blockchain (recall that we only consider $P2PKH$ transactions).

2. Next, the attacker selects a subset of the $SKList$ in all secret key space, which is used for verification.

3. The attacker takes the secret key from the $SKList$, generates the corresponding address, and checks whether the received address is inside the $AddrList$ set.

4. If it is, the attacker publishes a transaction that transfers the available funds from that address to the pre-generated address, this operation is repeated for all the secret keys in the SKList.

This attack is performed against all users with bitcoins, not against a specific user or address.

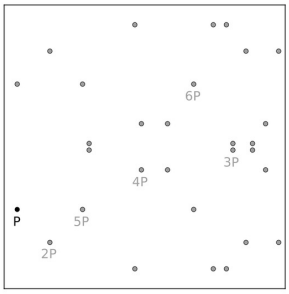
9.3 Example for small groups.

9.3.1 Who owns BTS?

> In a nutshell, except for some cases, anyone is the owner of a bitcoin if they can prove (following the BTC protocol) that they own the private key associated with the public address to which the bitcoin was previously sent.
>
> Thus, it is crucial for such a cryptosystem to generate private keys that do not allow their recovery from public information.

9.3.2 Attack on bitcoin ?

In the case of BTC, it depends on the complexity of the discrete logarithm elliptic curve problem ($ECDLP$) and the robustness of some hash functions to preimages, but if the keys are not uniformly distributed, there may be a (relatively) small portion of key space where they can be easily found. A bitcoin curve can be thought of as a set of scattered points within the Fp×Fp plane with a given base point P, where the multipliers form the entire curve.



Since private keys are $Zq*$ elements, a direct bot attack on the $Bitcoin$ system seems impossible because the inversion of the map:

$$\varphi : Zq* - \to E, k - \to k - P$$

implies the solution of the $ECDLP$ instance. However, there are several small subgroups:

$$H \leq Zq*$$

Which can be inspected, and for which an exhaustive calculation of all possible keys and corresponding addresses can be performed.

In this way the inverse of the bounded map can be computed:

$$\varphi|H : H - \to G$$

Since it is assumed that the keys are uniformly distributed, there is no probabilistic argument assuming they exist in certain small subgroups.

However, assuming that this is the case, we need to choose a suitable subgroup.

From this point of view, considering factorization:

$$q - 1$$

на простые целые числа :

$q-1 = 26\times3\times149\times631\times107361793816595537\times174723607534414371449‘¸p¸1x‘¸p¸2x\times34194848697416600052234$

it is not difficult to check that the maximal subgroup of moderate size (i.e., which can be checked with an average computer today) contains N elements, where:

$$N = 26 \times 3 \times 149 \times 631 = 18051648$$

Such a group can be easily obtained by considering any primitive element t of Zq, for example:

$$t = 7$$

and considering an element:

$$g = tp1 \times p2 \times p3$$

which generates a subgroup :

$$H =< g >= g^i | 1 \leq i \leq 18051648$$

Let F be a field, then any finite subgroup $G \leq F$ is cyclic. Moreover, for every positive integer M dividing $|G|$ there exists a single subgroup $H \leq G$ such that $|H| = M$

9.4 The Dangers of Quantum Computers.

Let's look into the future and talk about my (and hopefully yours) favorite quantum PCs. And how they help in brute-force attacks.

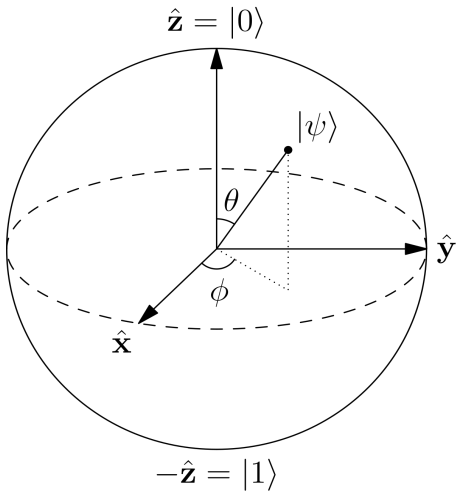
9.4.1 How do quantum computers work?

As we know quantum computers manipulate a subatomic state, called **superposition**, where a particle exists in several places at once until it is observed. Such particles in quantum pc are called **qubits**.

What is superposition?

> Quantum machines operate at near absolute zero, which allows them to operate beyond the binary ones and zeros found in a traditional laptop or PC. This means that quantum computers can perform multiple tasks simultaneously(Because qubits are in superposition, i.e. in all states at once, and simultaneously)

What are qubits?



> Cubits are atoms, ions, photons or electrons and their respective control devices, which work together to act as a computer memory and processor.

Because a quantum computer can contain these multiple states simultaneously, it can be millions of times more powerful than the most powerful modern supercomputers. This superposition of qubits gives quantum computers their inherent parallelism.

What does it give them?

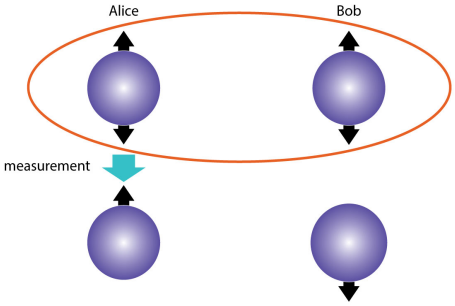
> This parallelism allows a quantum computer to work on a million calculations at once, while your desktop PC works on one. A

quantum computer with 30 qubits would be equal to the processing power of a normal computer, which can run at 10 teraflops (trillions of floating point operations per second).

Modern computers operate at speeds measured in gigaflops (billions of floating point operations per second). Today's typical desktop computers operate at speeds measured in gigaflops (billions of floating point operations per second).

Quantum computers also use another aspect of quantum mechanics known as entanglement

Quantum entanglement



This is the phenomenon that explains how two subatomic particles can be closely related to each other, even if they are separated by billions of light years of space. The strange part of quantum entanglement is that when you measure something about one particle in an entangled pair, you immediately know something about the other particle, even if they are separated by millions of light years. This strange connection between the two particles happens instantaneously.

If you take both quanta together, you find that there are correlations between the combined properties of both: something you couldn't know if you measured only one of them. Here's a good article for a basic **understanding of it**

Also, if you're interested in quantum mechanics, here's a cool course of **lectures on it**

9.4.2 What about bitcoin?

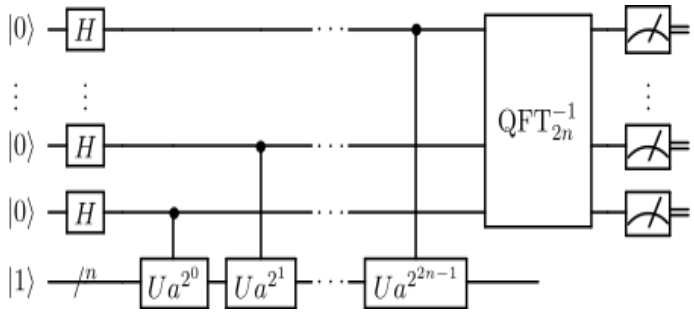
- As we remember in Bitcoin, data is recorded in transactions built from inputs and outputs.
- Each input refers to a previously unreferenced output and provides proof of authorization.
- Outputs that are not referenced, or "*unspent*" in a particular blockchain state, constitute a set of unspent transaction outputs ($UTXO$).
- Each output contains a call script that must be resolved by any input that wishes to refer to it.

- Most often the call script contains a hash of the public key pk, and in order to spend it, the input must provide this public key and a signature under the transaction that can be verified with pk.
- The act of providing this data serves as authorization to spend the output and further secures the transaction, as the signature can only be created with the secret key associated with the pk, which will never be revealed on the network.

In Bitcoin, cryptography is ECC as we know it, where the one-way function (trap) between the secret key and the public key is an exposition of elliptic curve points.

In fact, to derive the secret key from the public key, **it is necessary to solve the elliptic curve discrete logarithm problem** ($ECDLP$), for which no effective classical algorithm has been found (*Which is easy to solve one way, but would take many years to solve the other way*)

9.4.3 How does hacking work?



The **Shor algorithm** can be used to solve $ECDLP$ and many other forms of the hidden subgroup problem, which is the most pressing threat for most public-key cryptography in use.

It works by preparing a superposition of states, where each state is formed by concatenating x(secret key) with the value of $f(x)$ (the public key obtained by exponentiation).

- A **quantum Fourier** transform scheme can then be used to extract the period of the function f and then compute x for any givenu.

$$y = f(x)$$

- The Shor algorithm provides an attack on ECC and most public key cryptographies in polynomial time.
- When efficient QCs with large states are physically implemented, Bitcoin elliptic curve cryptography can be subverted using Shor's quantum algorithm.

In fact, an attacker with a quantum computer with about **1,500 qubits** can solve the $ECDLP$ and compute the ECDSA secret key given the public key.

Once the attacker figures out the secret key, he becomes indistinguishable from the original owner and can successfully sign transactions using any UTXO protected by that public key.

9.4.4 Public Key Disclosure.

For the purposes of this analysis, it makes sense to distinguish between slow QC and fast QC.

.

Let us first assume that a slow quantum computer is developed.

Although it can be used to solve the $ECDLP$, the computation takes longer than the time it takes to include the transaction in the blockchain.

Under this assumption, QC is able to derive a private key from a previously disclosed public key.

Bitcoin $UTXO$ with a $P2PK$ type call script displays the public key in the transaction results, although call scripts of this type are legacy, they currently secure about 1.77 million BTC.

Moreover, instances of public key disclosure can occur with any type of call script solution.

If the $UTXO$ of a protected pk is used, and the pk also protects other $UTXOs$ that are not referenced in that transaction, they will become vulnerable because the pk has been exposed.

About 3.9 million BTC are still stored in $UTXOs$ that are protected by public keys but disclosed in some other input.

In total, these two public key disclosures compromise at least 33 total BTCs.

Fast QC.

- We said before that only slow QCs are available, but the most powerful attack on Bitcoin can only be carried out with a fast QC, which can derive ECDSA secret keys faster than a new transaction is inserted into the blockchain.
- With this technology, an attacker can successfully intercept transactions in real time.
- Immediately after broadcasting the TX transaction, the attacker uses the public keys received on the TX inputs to compute the corresponding private keys.

- He then creates a second TX transaction that summarizes the same outputs as the original TX, but sends the funds to an address held by the attacker. Note that spending the same UTXOs is possible because the attacker controls all the private keys needed to generate valid signatures.
- If this procedure can be performed before TX appears on the blockchain, and TX has a higher fee, the miners would prefer to enable TX and cancel it as they profit.
- This attack is called real-time transaction hijacking because the original transaction is replaced with a malicious transaction, but the likelihood of success of such an attack depends on the performance of the CC, so the Bitcoin community will have plenty of time to defend against it.

Researchers estimated that a quantum computer with 1.9 billion qubits would take 10 minutes to break Bitcoin's encryption.

It would take a machine with 317 million qubits to do the same operation in an hour, but if you had a full day to try to crack the security, you would need a system containing only 13 million qubits.

Right now, the most powerful quantum computer developed by IBM boasts 433 qubits (as you understand this is very small).

We're obviously a long way from having machines with 13 million qubits available, and a machine with 317+ million qubits is a much better bet for practical bitcoin hacking.

Scientists at Sussex believe that with the rate of progress we are seeing now, sufficiently powerful quantum computers will not be realized within a decade, which brings us to the 2030 **years**.

9.5 Inst. for brute force .

In general, as I think, the most effective way to crack a Bitcoin wallet is hacking brute force password or pincode, because in contrast to the quantum PC there is not so much value, but still brute force attacks are very specific activities (without quantum PC), especially on private keys.

9.6 If you want to try your luck, here are some tools for doing so.

9.6.1 Bitcoin-wallet-cracker

Automated bitcoin wallet generator, which uses mnemonic codes and passphrases to search for wallet addresses.

9.6.2 BitcoinAddressFinder

- Free high performance tool to quickly scan random Bitcoin, Bitcoin Cash, Bitcoin SV, Litecoin, Dogecoin, Dash, Zcash private keys and search for addresses with a balance.

The main goal is to generate (bitcoin/altcoin) addresses as quickly as possible using the JVM in combination with OpenCL and check if the address (RIPEMD160 hash) was used before, which includes possible hash collisions.

Functions:

1. Support for blockchain addresses based on secp256k1.
2. Tested open source module (robust) that can be easily compiled yourself.
3. Vanitygen bitcoin addresses using regular expression pattern.
4. Runs completely offline, no internet required or used. You can run it in a bunker with a power generator and no one knows about it.
5. No need for synchronization to run multiple instances.
6. Random numbers are used and no search organization is required. Simply run on multiple computers. Check with a high-performance database containing addresses if the generated address is already in use.
7. Portable, platform independent, runs on the JVM.
8. Generate uncompressed and compressed keys simultaneously.
9. EC-Key generation via multiple CPU threads, multiple OpenCL devices (optional).

9.6.3 Bitcoin-bruteforce

The bitcoin public address brute force is written in Python with simplicity and speed in mind.

Functions:

1. Compare multiple wallets to increase hacking speed.
2. Split the workload across multiple CPU cores, multiple brute force functions.
3. Online wallet search (OBF).
4. Print generation output.

9.6.4 Bruteforce-wallet

The purpose of this program is to try to find the password to an encrypted Peercoin, Bitcoin, Litecoin wallet file (e.g. wallet.dat).

It can be used in two ways:

1. Try all possible passwords, taking into account the encryption.
2. Try all passwords in a file.

- There is a command line parameter for specifying the number of threads used.
- Sending USR1 signal to a running wallet brute-force process forces it to keep running.

9.6.5 Plutus

A bitcoin wallet collider that enumerates random wallet addresses.

- This program is essentially a brute-force algorithm, constantly generating random bitcoin private keys, converting the private keys into corresponding wallet addresses, and then checking the balance of the addresses.
- If a wallet with a balance is found, the private key, public key, and wallet address are stored in a *plutus.txt* text file on the user's hard drive.
- The ultimate goal is to randomly find a wallet with a balance out of 2160 possible existing wallets.32-byte hex strings are randomly generated by $os.urandom()$,and used as our private keys.
- The private keys are converted into corresponding public keys using the $fastecdsa$ python library. This is the fastest library to sign $secp256k1$. If you run this on Windows, fastecdsa doesn't support it, so we use $starkbank - ecdsa$ to create public keys instead.
- The public keys are converted into bitcoin wallet addresses using the standard binascii and hashlib libraries.

This project includes a pre-calculated database of every $P2PKH$ bitcoin funded address.

- The generated address is searched in the database, and if the address is found to have a balance, the private key, public key, and wallet address are stored in a *plutus.txt* text file on the user's hard drive.
- This program also uses multiprocessing via the $multiprocessing.Process()$ function to perform parallel calculations.

9.6.6 Btcbf

This is a fast and efficient bitcoin private key matching tool written in python.

- It works by generating random or sequential private keys and their corresponding public addresses.
- It then verifies the address through an onlineAPI or an offline database.

9.6.7 Private-keys

Brute force attack on bitcoin wallet private keys with 12 character passphrases.

Additional attacks and materiel.

Let's talk more about interesting attacks, and I will also attach interesting additional material. As we saw above, attacks depend on many factors, but I want to divide the vulnerabilities of bitcoin itself into 3 types.

10 Key generation vulnerabilities.

Vulnerabilities of key generation.

Materiel:

In the materiaal below I will tell you about bad generation attacks, but first I will tell you about a funny key generation vulnerability of Profanity hourly addresses

10.1 Profanity.

What is Profanity?

It is a tool for creating ambitious *Ethereum* addresses.

It generates addresses in parallel, using the power of a GPU with *OpenCL* using a simple algorithm.

10.1.1 Vulnerability.

This vulnerability is a random key generation.

To generate a random private key, Profanity first uses a random device ($std :: random\_device$ in $C++$) to generate an initial number.

But unfortunately, the initial value is $32-bit$, and it cannot be used directly as a private key.

To generate a $256-bit$ private key with self-written code $32-bit$ initial number into the pseudorandom number generator $mt19937\_64$, which is a deterministic function.

Simply put, you get a private key.

Since there are only 2^{32} possible initial key pairs ($d\_0,0$, $Q\_0,0$), and iteration at each step is reversible, you can crack the private key from any public key generated by $Profanity$.

The key storage vulnerability (key protection at rest) is what I wrote about above, talking about attacks on software and hardware wallets.

11 Transaction and blockchain vulnerabilities.

11.1 Lattice Attack.

11.1.1 ECDSA.

To begin with, the Elliptic Curve Digital Signature Algorithm ($ECDSA$) is a common digital signature scheme that we see in many of our code reviews.

It has some properties, but it can also be very vulnerable to private key recovery via a side-channel attack that reveals less than one bit of the secret one-time number.

We know what $ECDSA$ is and how it works, let us demonstrate its fragility.

11.1.2 Vulnerability.

Again, because this is a digital signature scheme, it is critical that the secret key is never disclosed to anyone but the person signing the message, but if the signer loses the signature as well as the one-time number used, an attacker can immediately recover the secret key.

Say I release the signature (r, s) for message m and accidentally discover that I used the one-time number k.

Since:

$$s = (k^{-1}(H(m) + xr))$$

we can easily figure out the secret key:

$$s = (k^{-1}(H(m) + xr))$$
$$ks = H(m) + xr$$
$$ks – H(m) = xr$$
$$x = r^{-1}(ks – H(m))$$

Consequently, the signer must not only keep their secret key secret, but also all of their one-time numbers they have ever generated.

Even if the signer keeps every single one-time $NONCES$ number secret, if he accidentally repeats a single one-time $NONCES$ number (even for different messages), the secret key can also be immediately retrieved.

11.1.3 Attack.

Let $(r, s1)$ and $(r, s2)$ be two signatures created on messages $m1$ and $m2$ (respectively) from the same one-time number, k - since they have the same one-time number, the values of r will be the same, so it is very easy to detect by an attacker:

$$s1 = k^{-1}(H(m1) + xr) and s2 = k^{-1}(H(m2) + xr)$$

$$s1 – s2 = k^{-1}(H(m1) – H(m2))$$

$$k * (s1 – s2) = H(m1) – H(m2)$$

$$k = (s1 – s2)^{-1}(H(m1) – H(m2))$$

Once we have recovered the one-time number k using the above formula, we can recover the secret key by performing the previously described attack.

11.1.4 Examples.

Here are some examples of implementing this.

Video:

Cool youtube video about this kind of attack

Article:

Here's a more selective article about it

.

Tool:

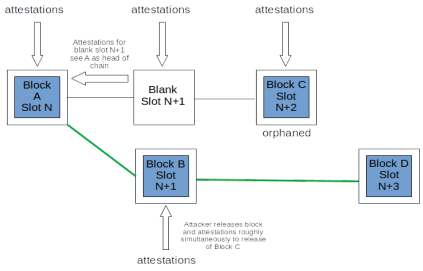
Here's an example of tools for such attacks

Functions:

Recover an $ECDSA$ private key from incomplete "k" one-time data.

Partial "k" information can be recovered from side channels.

11.2 Proof-of-Stake attacks.



11.2.1 What is Proof-of-stake.

- Proof-of-stake is a cryptocurrency consensus mechanism for processing transactions and creating new blocks in the blockchain.
- A consensus mechanism is a method of verifying records in a distributed database and securing the database.

In the case of cryptocurrency, the database is called a blockchain, so the consensus mechanism protects the blockchain. Proof-of-stake reduces the amount of computational work required to verify blocks and transactions. According to proof-of-work, it kept the blockchain secure.

> Proof-of-stake changes the way blocks are verified by coin owners' machines, so there is not as much computational work to be done.

Owners offer their coins as collateral, a bet, for the opportunity to verify blocks and then become validators.

Validators are randomly selected to validate transactions and verify block information.

This system randomizes who gets the fees, rather than using a mechanism based on competitive rewards such as proof of work.

Earlier versions of PoS typically used a fixed set of validators.

11.2.2 Problem.

This means that users who sold their shares at some point in the past can still participate in the checking process (in PoS checking is equivalent to chasing blocks) in the future.

This is a loophole because anyone who gets these obsolete keys (by rewriting history) can make a lot of money, this is also known as the nothing at stake problem. **Newer versions of PoS use dynamic validator sets and/or checkpoints to solve the problem.**

The idea is to revoke the right of past stakeholders to participate in future validation.

However, even with these measures in place, PoS protocols cannot completely solve the problem.

Two types of nodes remain particularly vulnerable:

1. New nodes that have just joined the network.
2. Long idle nodes.

Because these nodes either start with empty memory or have a huge gap in their memory, it will be difficult for them to detect that some stakeholder holders have already sold their coins in the "real" chain when they were not in the network.

Some PoS proponents will immediately point out that a similar problem exists for new nodes in PoW, so this is a problem.

The operator of a PoS node, even with the right software loaded, will have to regularly contact trusted third parties to make sure it remains in the canonical chain.

The fear of losing contact with the core network and getting into the wrong chain will go on forever, perhaps long after the trusted third parties have ceased to exist, indicating a significant deterioration in security.

All PoS protocols suffer from this problem.In this attack, the keys are current, which means that neither dynamic validator sets nor checkpoints will help.

11.2.3 Threats.

- An attacker with access to private keys that control at least 1/3 of the coin supply could easily create two equally valid blocks of identical height, with neither block looking more "correct" to the rest of the network.
- This is already an obstacle for most PoS protocols, as they are forced to terminate if the 2/3+ fair majority threshold is not met.
- No block can be "finalized" for any given stage. The PoS chain will stop in its tracks. So far in this analysis, we have assumed that in the worst case scenario, the attacker gains control of 1/3+ of the total number of coins, which is difficult but possible.
- But in fact, the requirements for the attacker are much lower, since the attacker only needs 1/3+ of the active share. It is unlikely that all coin holders will participate in the stealing and checking process.Let's say the participation level is 50%, then the attacker only needs 1/6 of the coin stock to trigger conflicting blocks/control points, instead of 1/3.
- If the participation level is 25%, the attacker only needs 1/12. This is quite disturbing, because a handful of the richest shareholders could easily control 1/12 coins. The low level of participation in stacking, is the biggest threat to PoS protocols.

11.2.4 Another problem .

Another problem that exacerbates the problem of current key theft.This is the requirement to connect to PoS:

1. Validators must store their stacking keys on the network in order to sign transactions. The fact that these keys are always connected to the network means they are much more vulnerable to hacking or theft.
2. Whether or not these staking keys directly control the funds behind the stakes, those who gain control of most of these staking keys (belonging to the active validator set) will gain control of the minted blocks.

12 Transaction vulnerabilities.

12.1 What is the vulnerability?

This is an old vulnerability, but it still exists.

Not many bitcoin companies/wallets reuse values when signing transactions these days, but people who create new copies of old coins and wallets are usually unaware of this vulnerability.

In researching this, I found that many Russian bitcoin hackers have coded bots to automatically grab coins from vulnerable addresses of this and other types, as mentioned at the beginning of this article.

Here are a few ways a bitcoin address or wallet can be vulnerable:

1. A private key is created with a shared password, such as "123456."
2. A simple copy/paste error.
3. Transaction is created with non-standard outputs, random number generator was not used correctly or gave the same result.
4. The private key was published publicly.

12.2 We will talk about a transaction with a broken random number generator(string).

These addresses reuse certain values in the transaction because of bad knowledge, programming errors or a broken random number generator.

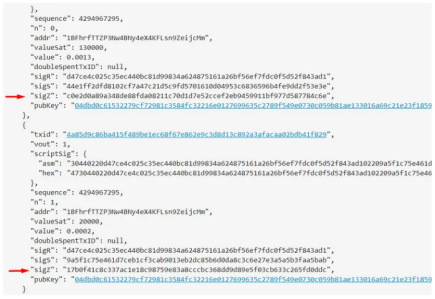
- The beginning of the scripts contains captions (defined as $'r'$ and $'s'$).
- The end of the script is the hexadecimal public key.

If r values in the scripts are exactly the same - it means that we can get the private key.

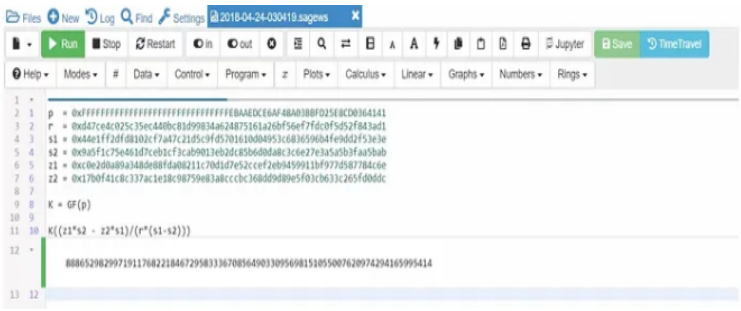
$$BitcoinPrivateKey = (z1 * s2 - z2 * s1)/(r * (s1 - s2))$$

12.2.1 How to get a private key.

- We have the values of r and s, now we need to find the values of $z1$ and $z2$.
- Type in your transaction ID at `https://2coin.org/` and scroll down to find the z values.



- We will need to create a final field for the calculation.
- You can use *Sagemath*: `http://www.sagemath.org/`



- Once you get the results of the calculations, you need to convert them from decimal to hexadecimal.
- You can do this here: `https://www.rapidtables.com/convert/numbo-hex.html`

- From there you can convert it to WIF and it will all be a private key.

Here are more detailed articles:

- `https://dawood.me/how-i-hacked-a-bitcoin-wallet-a-step-by-step-guide-da03cbccba39`
- `https://strm.sh/studies/bitcoin-nonce-reuse-attack/`

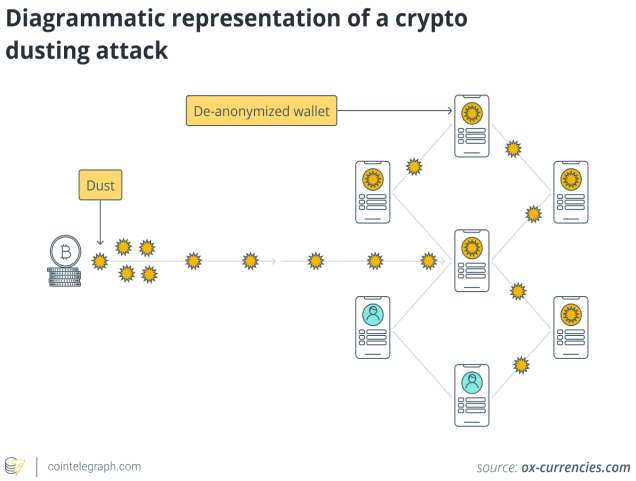
12.3 Conclusion.

Many people think:

- "*what difference does it make what number generator and how to use it?*"
- "*use the library generator in my programming language without even reading the documentation, and everything will be fine.*"
- "*no one would think of breaking it.*"

But as we can see this is not a small problem, because not even a particularly advanced attacker, with minimal resources can get your secret key and steal all the money. That is why I ask you to be very serious about all moments of key generation, including not being lazy and choose a good random number generator with high entropy, and use it correctly, to prevent such trivial, but at the same time fatally serious problems.

13 Dust Attacks.



13.1 What is "*dust*".

In the language of cryptocurrency, the term "dust" refers to a tiny number of coins or tokens - a number so small that most users don't even notice. Taking Bitcoin as an example, the smallest unit of BTC is 1 satoshi (0.00000001 BTC), so we can use the term "dust" to refer to a couple of hundreds of satoshi.

13.2 Attack.

13.2.1 Why such attacks are possible.

Attackers realized that cryptocurrency users don't pay much attention to those tiny amounts displayed in their wallet addresses, so they started "*dusting*" a large number of addresses by sending a few satoshis (i.e. small amounts of LTC, BTC or other cryptocurrency) to them.

- After clearing different addresses, the next step of the attack involves a combined analysis of these addresses in an attempt to determine which ones belong to the same cryptocurrency wallet.
- UTXO in bitcoins is best viewed as a set of coins (for smaller amounts) and bills (for larger amounts).
- When you make a payment, there is a very good chance that you will give the seller a combination of smaller coins or a bill for which you will get change.
- There is also a chance that you are not keeping track of the change amount in your wallet.

- Abusers know this and take advantage of it by "cleaning out" your wallet.

13.2.2 How Attacks Happen.

To continue the analogy (obviously fiat is several orders of magnitude more fungible than bitcoin, so accept the assumption that each coin has a unique number), imagine that an attacker decides to make an exact copy of a coin, even though there is some camera/scanner built in which reads and identifies your other coins when you make a payment.

In the case of Bitcoin, the attacker sends dust to a given address and waits for it to be used in conjunction with other UTXOs, thereby discovering which addresses also belong to the target.

Forced, address reuse:

- This is when an attacker pays a (often small) amount of bitcoins for addresses that have already been used in the blockchain.
- The attacker hopes that the users or their wallet software will use the payments as input for a larger transaction that will identify other addresses through experimental shared-entry.
- These payments can be seen as a way to force the owner of the address to unintentionally reuse the address.

13.3 Additional Matter.

Article:

> Here is an article with the implementation of such an attack: https://dust-attack.blogspot.com/2019/03/blog-post.html

Examples of attack tools:

1. `https://github.com/ZenGo-X/big-spender`
2. `https://github.com/petertodd/tx-flood-attack`
3. `https://github.com/mloporchio/DustAnalysis`

14 Traffic analysis.

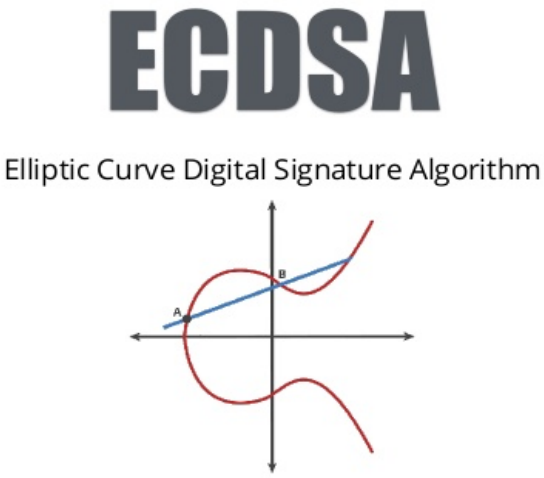
14.1 How and when an attack occurs.

Some organization listening in on your traffic may see you communicating on the bitcoin network.

- If an attacker sees a transaction or block coming out of your node that was not previously logged in, they can almost certainly say that the transaction was made by you or the block was mined by you.

- Because Internet connections are involved, the attacker will be able to link the IP address to the discovered bitcoin information.
- An attacker who cannot listen to all Internet traffic, but has many Bitcoin nodes to stay close to the sources, can learn the IP address that announces transactions or blocks.
- In addition, some wallets periodically rebroadcast their unconfirmed transactions so that they are more likely to spread widely across the network and be mined.

15 Attack on the ECDSA.



15.1 What is the attack?

The attack looks at the fact that you can always define the recurrence relation between the one-time numbers used in different ECDSA signatures as a polynomial of arbitrarily high degree with unknown coefficients modulo order of the point forming the curve.

- If you have a set of N $ECDSA$ signatures (for the same private key) and this recurrence relation has degree D, then (with some caveats, which we will discuss later) you can use the $ECDSA$ signature equation to rewrite the polynomial into the private key terms and the unknown recurrence factors.

Researchers have found that the unknown coefficients can be excluded from a polynomial whose roots always include the signing party's private key.

So, if D is small and you have enough such correlated signatures

$$(N \geq D + 3)$$

then you can perform a key recovery attack simply by finding the roots of a polynomial with known coefficients over a finite field, which is a simple task for a PC!

15.2 Launching an attack.

To launch an attack in practice requires:

1. At least 4 signatures
2. Generated by the same private key, corresponding public key and message hash associated with each signature.

To launch an attack in practice requires:

1. A minimum of 4 signatures generated with the same private key
2. A corresponding public key and message hash associated with each signature.

If the one-time numbers obey the recurrence relation, we extract the private key used to create the vulnerable signatures.

- The more signatures used in an attack, the slower it becomes, but the greater the probability of success.
- If you attack N signatures and their N one-time numbers, they follow a recurrence relation of degree no higher than $N - 3$, and you can perform a key recovery attack on ECDSA.
- The attack means that every time an $ECDSA$ signature is generated, the signature itself gives us the link between the one-time number and the private key.

15.3 Weak PRNGs.

If one-time numbers are indeed randomly generated, this should never be a problem, because the probability that the number of randomly chosen one-time numbers corresponds to a low-degree polynomial recurrence relation is negligible, but there is a catch:

- One-time numbers are usually output by a pseudorandom number generator (PRNG), and PRNGs are deterministic algorithms with relatively low complexity.

- At best, the PRNG used is sufficiently complex and cryptographically secure - this means that any polynomial correlation between its outputs will have an astronomically high degree.
- You can safely consider it indistinguishable from truly random, but weak PRNGs are almost everywhere.
- Take, for example, the simple case of a linear congruent oscillator (LCG), which is a typical textbook introduction to implementing PRNGs.
- The LCG for PRNGs is the same as ROT13 for encryption and 1234 for password protection.

15.4 Additional material.

Here is an example of such an attack

16 Chosen-IV attack on bitcoin.

16.1 Chosen-IV Attack

16.1.1 Streaming Ciphers.

Streaming ciphers combine a secret key with a matched initialization vector (IV) to create a pseudo-random sequence that is re-synchronized from time to time.

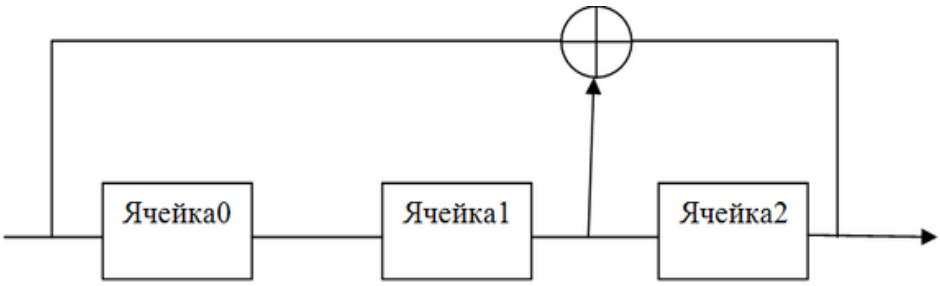
16.1.2 Vulnerability.

The Chosen-IV attack on bitcoin relies on the detection of specific IVs that will reveal information about the secret key. Usually several pairs of IVs are selected, and the differences in the generated key streams are then analyzed statistically for linear correlation and/or algebraic logical relation.

- -f the selection of certain initialization vector values does reveal a non-random pattern in the generated sequence, this bitcoin attack computes some bits and thus reduces the effective key length.
- A symptom of a bitcoin attack would be frequent resynchronization.

Modern stream ciphers include steps to adequately mix the secret key with the initialization vector, usually by performing many initial steps.

16.1.3 Streaming Cipher.



The stream cipher :

- This is a symmetric key cipher in which the plaintext digits are combined with a stream of pseudo-random cipher digits.
- In a stream cipher, each plaintext digit is encrypted one at a time with the corresponding key stream digit to produce a ciphertext stream digit.
- Because the encryption of each digit depends on the current state of the cipher, it is also known as a state cipher. In practice the digit is usually a bit, and the join operation is an exclusive xor operation.

16.2 Attack.

Bitcoin is based on the detection of specific IVs that will reveal information about the secret key.

Usually, several pairs of IVs are chosen, and the differences in the generated key streams are then analyzed statistically for linear correlation and/or algebraic logical relation (see also Differential Cryptanalysis).

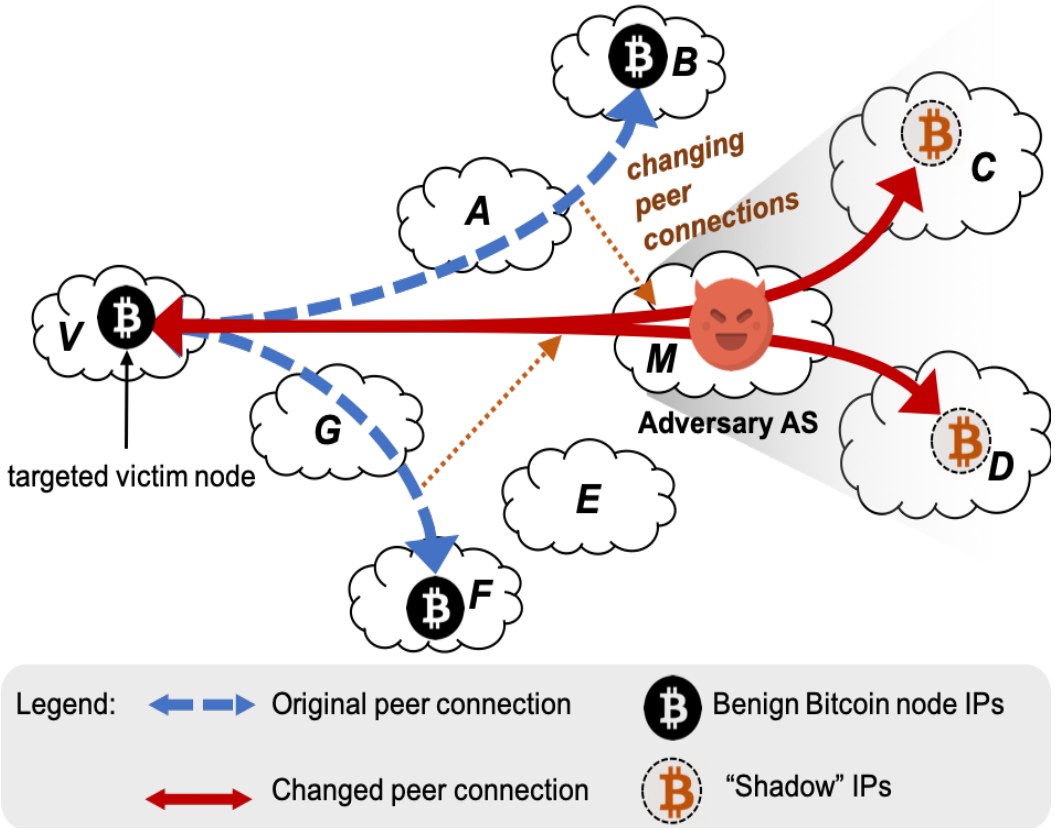
If the choice of certain initialization vector values does reveal a non-random pattern in the generated sequence, this bitcoin attack computes some bits, and thus reduces the effective key length.

A symptom of a bitcoin attack would be frequent resynchronization.

Modern stream ciphers include steps to mix the secret key with the initialization vector, usually by performing many initial steps.

Not a bad article on this, with the addition of encryption

17 Partitioning attacks.



17.1 Attack Protection.

- Transaction verification requires that nodes know the ownership and balance of each Bitcoin address.
- All of this information can be derived from the Bitcoin blockchain, an authenticated data structure that effectively generates a ledger of all accepted transactions.
- Bitcoin is able to synchronize the blockchain in an asynchronous way, and attackers can attempt to disrupt this process.

17.1.1 Synchronization Values

Synchronization is crucial:

- Conflicting transactions attempting to transfer the same bitcoins to different destinations can be approved by miners who are unaware of each other.
- Blockchain bitcoin blockchain creation consists of blocks, batches of transactions that are sequentially added to the ledger.
- Each block contains the cryptographic hash of its predecessor, which determines its place in the chain, and a proof-of-work.
- The proof-of-work serves to make it harder to create a block and reduces the number of conflicts in the system.
- Conflicts that take the form of blocks, those in turn extend the same parent block, are alternative sets of accepted transactions.
- Nodes converge to a single consistent version, choosing the chain containing the most computational work as the accepted version (usually the longest chain).

17.1.2 How this is useful.

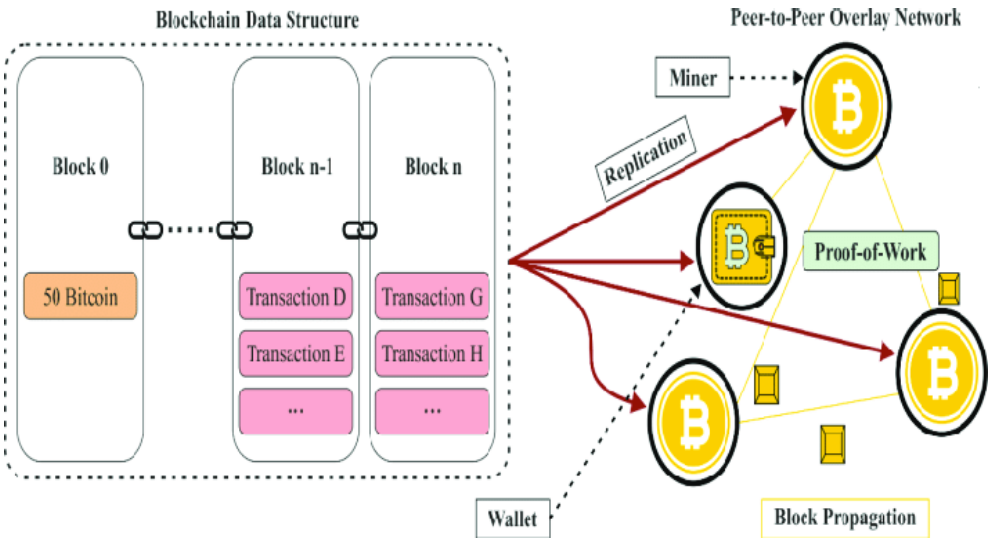
It limits an attacker's ability to compromise the system:

1. they cannot easily create a large number of blocks, potentially allowing them to create a longer alternate chain that accepts nodes and thus cancels the transfer of funds (double spend). -
2. The complexity of block creation is arranged in such a way that one block is created in the network every 10 minutes on average.

This is done to give blocks enough time to propagate through the network.

- However, if delays are high compared to the speed of block creation, many forks are created in the chain as blocks are created in parallel.
- In this case, the number of discarded blocks (known as "orphan" or "fork" speed) increases and protocol security is degraded.
- Newly created blocks are propagated through the network using the gossip protocol.
- In addition to block propagation, nodes also propagate transactions among themselves that await inclusion in the chain by that node, and then the next block is created.

17.2 How Bitcoin Works.



Bitcoin acts as a peer-to-peer network with each node maintaining a list of IP addresses of potential peers. The list is loaded via a $DNS - server$, and additional addresses are exchanged between peers. By default, each node randomly initiates 8 unencrypted $TCP-connections$ with identical coadditions in different prefixes /16.

- Nodes additionally accept connections initiated by others (by default on port 8333).
- The total number of connections that nodes can establish is 125 by default.
- Nodes continuously listen for block announcements, which are sent via $INV - messages$ containing the hash of the announced block.
- If a node determines that it does not have a new declared block, it sends a $GETDATA$ message to a single neighbor. In response, the neighbor sends the requested information in a $BLOCK$ message.
- Requested blocks that do not arrive within 20 minutes cause the neighbor to disconnect and be requested from another.
- Transaction propagation occurs with a similar sequence of $INV, GETDATA, TX$ messages in which nodes announce, request and exchange transactions that have not yet been included in the blockchain.

17.3 Attack.

17.3.1 Target.

- The goal of a separation attack is to completely disconnect a set of nodes from the network.

17.3.2 What is required?

- This requires the attacker to reroute and sever all connections between this set of nodes and the rest of the network.
- The attacker can reliably isolate the selected set of nodes using BGP intercepts.

This procedure is practical and requires only a basic knowledge of Bitcoin topology, namely the $IP-addresses$ of the nodes the attacker wants to isolate. Due to the complexity of the Bitcoin network (e.g., pools with multiple addresses and secret peer-to-peer agreements between pools), the initial isolated set may contain nodes and information leaks.

17.3.3 Process.

The attacker first redirects traffic destined for nodes in P by intercepting the most specific prefixes hosting each $IP-address$, the attacker intercepts Bitcoin traffic (e.g., based on $TCP-ports$) and determines the corresponding partition connections it attempts to create.

If this is the case, the attacker discards the packets, if not, meaning the connections are within P, it monitors Bitcoin messaging to detect "*leakage points*".

- Leak points are nodes currently within P that maintain connections to nodes outside P, and it is the "*hidden*" connections that the attacker cannot intercept. The attacker can detect these nodes automatically and isolate them from other nodes in P.
- Eventually, the attacker isolates the maximum number of nodes in P that can be isolated.

17.3.4 Example of an attack.

Suppose there are two mining pools depicted as grass.

- Both pools are multihomed and have gateways in different $ASes$.
- For example, the first pool has gateways located in $AS4$, $AS5$ and $AS6$.
- Any AS on the connection path can intercept connections within the pools.

Consider the $AS8$ attack, which aims to isolate the set of nodes

$$P = (A, B, C, D, E, F)$$

First, it intercepts the prefixes advertised by

$$AS1, AS2, AS6$$

as they place nodes within P attracting traffic destined for them.
Next, $AS8$ rejects all connections crossing a partition: those.

$$(A, J), (B, J), (F, G)$$

Note that node F is inside the isolated set P, but the gateway that was redirected by the intercept.

Most likely F is exchanging data with it.

This connection may not be based on the *Bitcoin* protocol, so it cannot be intercepted (at least not easily).

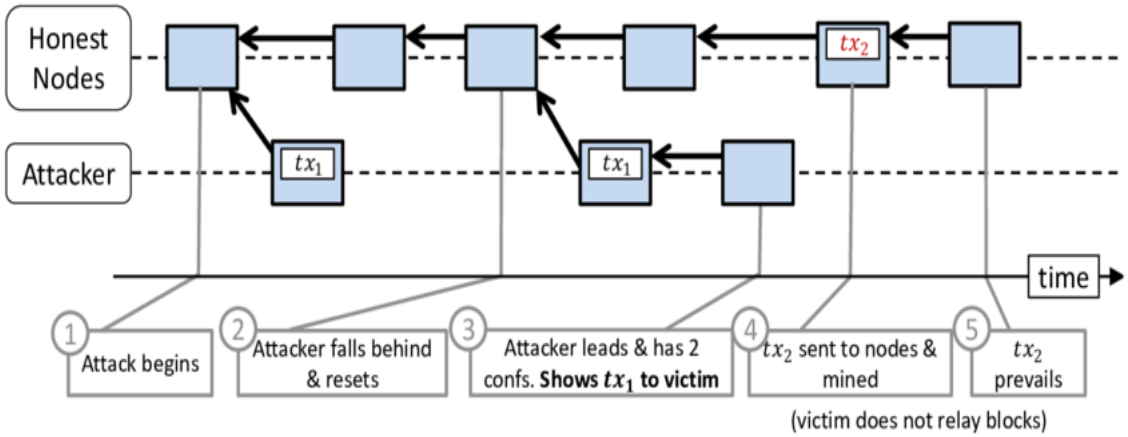
Thus, even if an attacker discards all *Bitcoin* connections that he intercepts, node F can still learn about the transactions and blocks made on the other side.

It is not possible to isolate P as such, but $AS8$ can determine that node F is the point of leakage during the attack and exclude it from P, effectively isolating

$$I' = (A, B, C, D, E)$$

instead, I' is the maximum subset of P that can be isolated from the Bitcoin network.

18 Attack Vector76.



18.1 How the attack works.

18.1.1 Methods.

Vector76 is a combination of the Race attack and the Finney attack.

1. In this case the attacker creates two nodes, one connected only to an exchange node and the other connected to well-connected nodes in the blockchain network.
2. The miner then creates two transactions, one with high value and one with low value.
3. The attacker then pre-mines and withholds the block with the high value transaction from the exchange service.
4. After the block is announced, the attacker quickly sends the pre-mined block directly to the exchange service.
5. He, along with some miners, will treat the pre-mined block as the main chain and confirm that transaction.

Thus, this attack exploits the fact that one part of the network sees the transaction that the attacker included in the block, while the other part of the network does not see the transaction. After the exchange service confirms the high value transaction, the attacker sends the low value transaction to the main network, which rejects the high value transaction.

18.1.2 Result.

The result is that the amount of the high value transaction is credited to the attacker. Although this type of attack has a high chance of success, it is not common because it requires placing an electronic wallet that accepts payment after one confirmation, and a node with an incoming transaction.

19 Additional materials.

19.1 Crypto-attacks

This is a good compilation on crypto attacks.
Realized attacks:

- Shamir's Secret Sharing.
- RC4.
- RSA attacks.
- One-time Pad.
- Mersenne Twister.

- Approximate Common Divisor.
- CBC.
- ElgGamal Signature.
- Other interesting implementations.
- Shamir's Secret.
- Small Roots.

19.2 Blockchain-and-crypto-currencies

Cool article on github about methods of attacks on bitcoin and blockchain, which explains how bitcoin, transactions and network work.

The article also gives examples of an attack on:

- $UTXO$ address change detection.
- Transactions and more.

19.3 Bitcoin-nonce-reuse-attack

A bit of material on reused transactions.

- When a one-time number is reused, it becomes vulnerable and leads to the solution of a trivial equation.
- This vulnerability is related to incorrectly selecting a random number when signing a message in the $ECDSA$ system. A similar vulnerability was discovered on PS3 in 2010, in 2013 a hacker, knowing about it, launched an attack on Bitcoin transactions.
- Due to a bug in $PRNG$ *Android*, it did not initialize correctly, transactions with more than one input reused the same, random value for the signature, which could be detected simply by finding a duplicate p value in the transaction signatures.

The bug was detected in the $SHA1PRNG\_SecureRandomImpl$ class when the $engineNextBytes$ function in the $SecureRandom$ implementation of *Apache* $Harmony$ $6.0M3$, used in the *Java* Cryptography Architecture (JCA) in *Android* before version 4.4 and other products, was not initialized correctly, using the wrong offset value, making it easier for attackers to break cryptographic security mechanisms using the resulting $PRNG$ predictability.

19.4 FinderOuter

A bitcoin recovery tool that simplifies the recovery process for anyone with any level of technical knowledge. It uses a simple user interface with a list of recovery options. Each option has an explanation and many hints to help the user understand what is needed.

- It always consists of filling in some text boxes and selecting some options and pressing the Find button.
- This saves the user from having to read long pages of a manual on how to use the application. Each option also has several example cases that can show a simple preview of how each option should be filled out for different cases.

19.5 Physical-bitcoin-attacks

A list of known attacks on bitcoin/crypto-asset targets.
This list is not exhaustive, many attacks are not publicly reported.

19.6 CryptoAttacks

Executing attacks on cryptosystems, which describes the following attacks:

- Clustic attacks: $One\ time\ pad/xor$.
- Block attacks: CBC, ECB, GCM, $Whitebox\ AES$. - Public Key attacks: **RSA**.
- Elliptic curve attacks: $ECDSA, Hash, PRNG, Utils$.

19.7 Remote-timing-attacks-are-practical

- Attempting the "*Remote Timing Attacks are Practical*" attack on OpenSSL

19.8 Bitcoin-lost-deep-key-private-key

Did you know that it is possible to scan bitcoin private keys that are online because of bitcoin development vulnerabilities, because there is nothing that really protects the internal structure of a bitcoin wallet in either the classic or basic or other variants.

Even when encrypting your wallet, there is nothing to prevent you from importing your private key by changing the configuration flag in any wallet, whether classic or basic or variant.

The author tried to test a basic attack **vector using a JTR and crunch** based password generation program to create a range of values with base 58 based on bitcoin private keys, and with further modification you can continue to customize this form of deep private key scanning by running it as a basic attack vector.

As an example of bitcoin network private key generation and experimentation, some private keys start with $\#L5...c4$ or $Ky...k1 <$ as an example 52 alphanumeric characters without special characters, decoded they flip to an address starting with 1 as part of the final decoding process.

19.9 Crypton

This is an educational library for learning and practicing offensive and defensive cryptography. It is essentially a collection of explanations and implementations of all existing vulnerabilities and attacks on various encryption systems (symmetric and asymmetric), digital signatures, message authentication codes and authenticated encryption systems.

There's a materiel on:

- RSA encryption.
- PKCS1-v1.5 encryption with RSA.
- Attacking the selected ciphertext with RSA cryptosystem.
- Oracle LSB decryption.
- Oracle attack on RSA PKCS1 augmented encryption system.
- Ferm factorization. - Sieve enhancement. - Copperman's attack. - Bone Derfee attack. -
- Hastad's broadcast attack.
- HBA implementations for augmented messages. - Pseudorandom number generators.
- A linear congruent generator.
- Mersenne Twister PRNG.
- Shift register with linear feedback.
- Digital Signatures, Blinding attack on RSA digital signatures.
- Diffie Hellman key exchange.
- Elliptic Diffie Hellman curve.

19.10 Cryptocurrency-Security-Audit-Guide

A guide to blockchain-based cryptocurrency security auditing.

It can be used to model cryptocurrency threats.

It also includes different testing methods (black box testing). It tells about vulnerability severity and different attacks (attacks on applications or smart contacts).

19.11 Identifying-Key-Leakage-of-Bitcoin-Users-Brengel-Rossow

Interesting article about key leakage analysis.

- It looks at the problem of key leaks.
- For this, researchers track the *Pastebin* channel from September 2017 to March 2018 to find public bitcoin secret keys, showing that attackers may have stolen 22.40 BTC worth approximately $178,000$.
- The article then describes the key leak,with incorrect use of cryptographic primitives, and scanning the Bitcoin blockchain for reuse of the one-time $ECDSA$ number.
- The researchers described how an attacker could use duplicate r values to leak one-time numbers and secret keys, which goes beyond the standard cases where the same one-time number and the same key were used in combination more than once.
- Their results show that the reuse of the one-time $ECDSA$ number has been a persistent problem in the Bitcoin ecosystem and has already been used by attackers. In fact, an attacker could have used the reuse of nonce to steal 412.80 BTC worth approximately $3.3 million.

19.12 Bitcoin-hacking-tools

The source code of the main tools used in bitcoin attacks, where you will find:

- Brutforce Wallet.
- Wallet bruteforcing.
- ECDSA private key recovery.
- Big bitcoin collider script.

19.13 Blockchain-Attack-Vectors

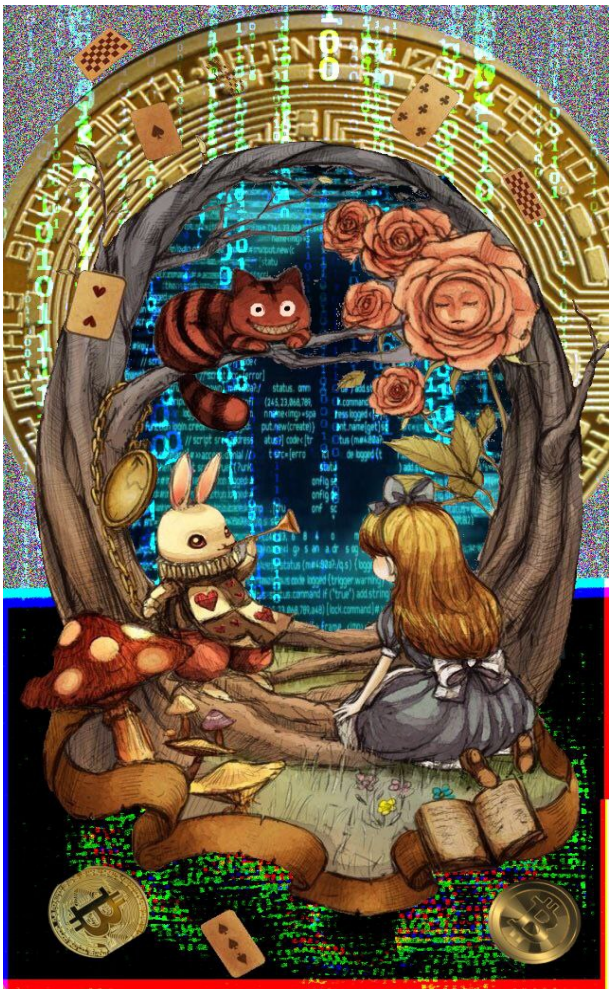
Blockchain attack vectors and smart contract vulnerabilities. In this article we will talk about all known blockchain attacks and smart contract vulnerabilities. Blockchain is not really as secure as we think, even though security is integrated into all blockchain technologies, even the most secure blockchains are subject to attacks by modern Cyber criminals.

19.14 Twist_attacks

An article about attacks on elliptic curves and their dangers, with a brief introduction to the curves themselves (closure and public keys, etc.).

- It also describes twist attacks on secp256k1, and gives the Chinese theorem on residues and details of the Twist attack, talks about reconstructing the Bobs private key, the coolest part - there are code examples. The author sums it all up by showing how to avoid this kind of attack.

20 Conclusion.



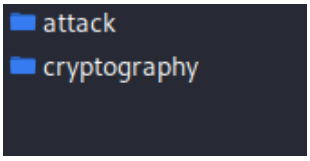
Having done a lot of work and trying to convey to you the whole essence of cryptocurrency cryptography I made the following conclusions:

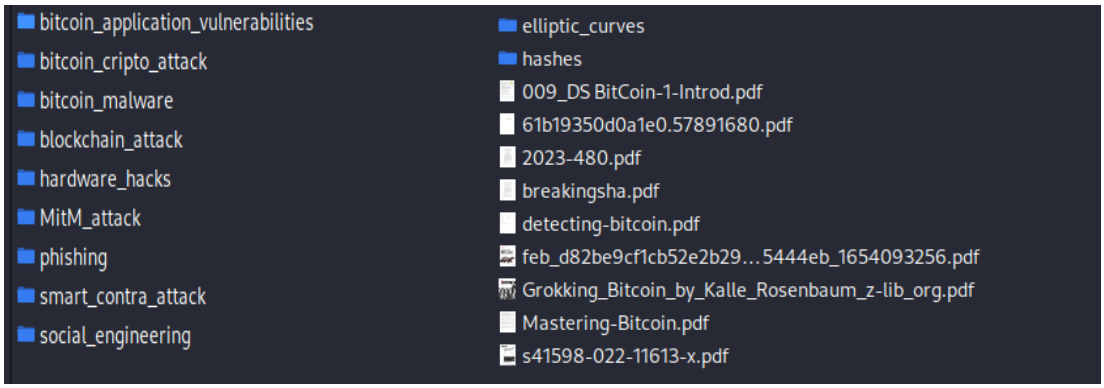
- - Many people (especially those who haven't read my article☺) think that bitcoin wallets can't be hacked, it's easier to visit a wallet holder with a bat and using physical force make him transfer bitcoins to his wallet, otherwise "*it will take millions of years to brute force bitcoin keys*".
- - As we can see (after reading my article) anything is possible to hack, and often it is not as hard as it seems.
- - In this article we touched hacking depending on human factor (social engineering), hacking web wallets, hardware wallets, blockchain network itself, and we saw that even cryptography can be attacked, and quite successfully ☺.
- - Some of the attacks are easy to implement, some are hard to do, some (attacks using quantum PCs) will only be available in the future, nevertheless all of them are feasible.

20.1 What next?

- I'm not saying that this article contains all material about attacks on wallets, on the contrary, it's a very big article and I omitted a lot of them, deciding to describe the most interesting things for my subscribers. And a lot of material about serious attacks on blockchain is not publicly available.
- I tried to cover this topic as much as possible (as much as possible for an article of this kind) to help you poke your head down this rabbit hole, and it is your choice whether to go down it or not. ✠
- I hope I succeeded in making you discover something new, and take one more step towards a magical wonderland.ƪ

20.1.1 Archive.





If you want to go deeper into this topic, I advise you to do your own research, there are many useful materials in the archive below, in my opinion:

- Cryptography Articles.
- Articles about cryptographic attacks
- Articles about viruses.
- Articles about social engineering.
- Hacking hardware wallets

Recommended reading.

20.2 The End.

At this point our journey through the wonderful world of bitcoin wallets comes to an end, but you can continue your journey without me.

Thank you for reading☺⚹.

And remember, the Hare and the Mad Hatter insulted time itself, so throughout the day drinking tea, thus shortening time, so my young (or not so young) Alice be vigilant and manage your time wisely, while not forgetting about the safety of your crypto wallets. ✠★