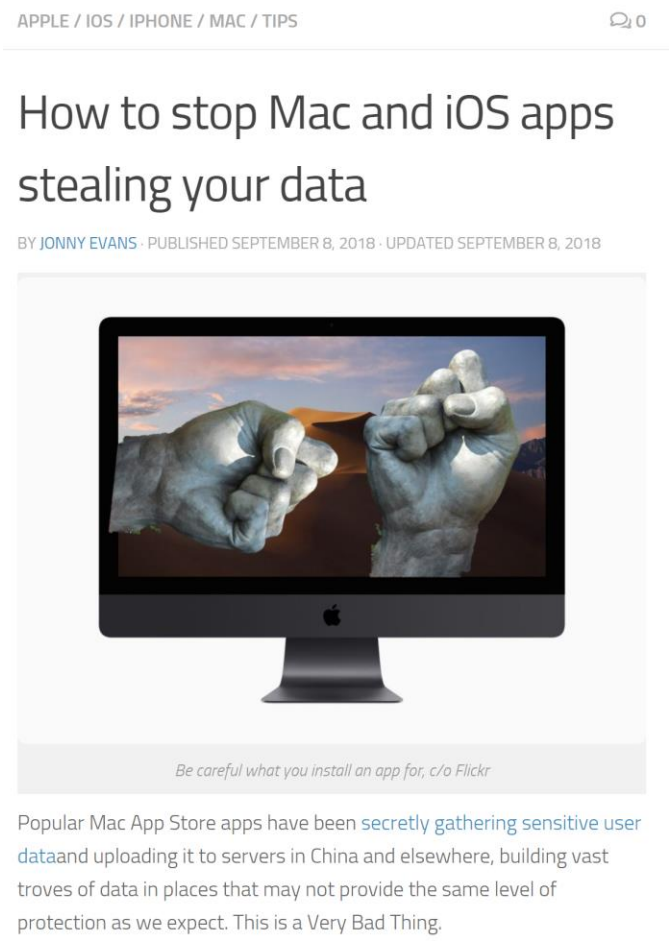
CO 445H

BLOCKCHAIN SECURITY

Dr. Benjamin Livshits

Apps Stealing Your Data



APPLE / IOS / IPHONE / MAC / TIPS 0

How to stop Mac and iOS apps stealing your data

BY JONNY EVANS · PUBLISHED SEPTEMBER 8, 2018 · UPDATED SEPTEMBER 8, 2018

Be careful what you install an app for, c/o Flickr

Popular Mac App Store apps have been secretly gathering sensitive user dataand uploading it to servers in China and elsewhere, building vast troves of data in places that may not provide the same level of protection as we expect. This is a Very Bad Thing.

What are they doing with this data?
We don't know what is happening with this data once it is collected. It's conceivable that this information could be analysed alongside other collections of data to provide insights into a person's identity, online activity, or even political beliefs. Cambridge Analytica and other dodgy behavioural modification companies taught us this.

The fact is we don't know what is happening to the data that is being exfiltrated in this way. And in most cases we are not even aware this is taking place.

The only reason we know about this collection of data-stealing apps is because security researcher, Patrick Wardle told us. Sudo Security Group's GuardianApp claims another set of dodgy privacy eroding iOS apps, while Malwarebytes has yet another list of bad actors.

http://www.applemust.com/how-to-stop-mac-and-ios-apps-stealing-your-data/

From Malwarebytes

https://objective-see.com/blog/blog_0x37.html

Objective-See products blog talks malware about

A Deceitful 'Doctor' in the Mac App Store

a massively popular app, surreptitiously steals your browsing history

09/07/2018

Our research, tools, and writing, are supported by "Friends of Objective-See"
Today's blog post is brought to you by:

Malwarebytes

become a friend!

Updates:

- The application, "Adware Doctor" has now been removed from the Mac App Store!
- I've uploaded the app's binary if you want to play along (download: **Adware Doctor.zip**)
- In Mojave, the sandbox will (always) protect private content, such as Safari's history.
- While process enumeration is disallowed in the iOS sandbox, and yes, /bin/ps is blocked on in the macOS sandbox as as well, Apple has noted that sandboxed apps may still enumerate running processes (though this will likely change in the future).

Background

You probably trust applications in the Official Mac App Store. And why wouldn't you?

Apple states:

> "The safest place to download apps for your Mac is the Mac App Store. Apple reviews each app before it's accepted by the store, and if there's ever a problem with an app, Apple can quickly remove it from the store."

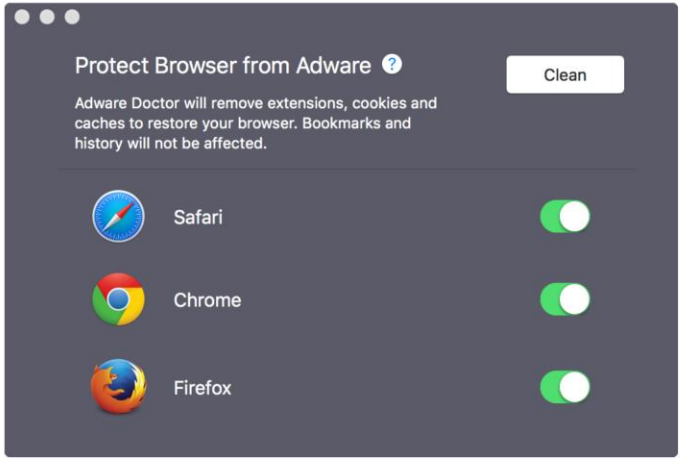
However, it's questionable whether these statements actually hold true, as one of the top grossing applications in the Mac App Store surreptitiously exfiltrates highly sensitive user information to a (Chinese?) developer. Though Apple was contacted a month ago, and promised to investigate, the application remains available in Mac App Store even today.

Note:
The nefarious logic of the app was originally uncovered by **@privacyis1st**. So major kudos to him!

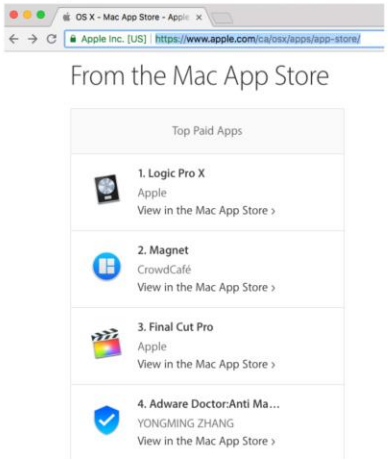
After he reached out, we collaboratively investigated this issue together.
#TeamWork

Adware Doctor

Adware Doctor claims to be the "best app" to remove a variety of common adware threats which target Mac users:



Found in the official Mac App Store, the application is massively popular. It is a top grossing application sitting at spot #4 ...meaning it is listed on Apple's main website!



Did You Just Steal My Browser History!?

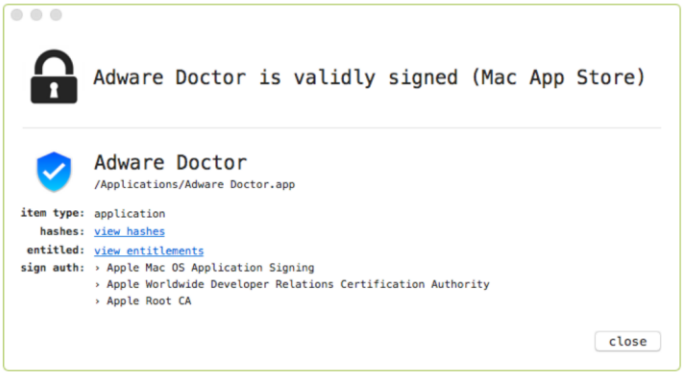
Did You Just Steal My Browser History!?

In a recent tweet, **@privacyis1st** noted that: "*`Adware Doctor` is stealing your privacy*":

In this tweet, he posted a link to a **video**, that appeared to show `Adware Doctor` collecting and stealthily exfiltrating a variety of sensitive user data including browser history. Yikes!

So, let's dive in a see what's going on! We'll use a combination of static analysis (disassemble) and dynamic analysis (network monitoring, file monitoring, & debugging) we get clear picture of what's going and ultimately confirm **@privacyis1st**'s worrisome findings!

First, let's **download** `Adware Doctor` from the official Mac App Store ...yes, paying the $4.99. We can confirm the application (as is the case with all applications in the Mac App Store) is validly signed by Apple proper:



Adware Doctor Stealing Browsing History

https://vimeo.com/288626963

Blockchain without the Hype

- Distributed ledgers and blockchain specifically are about establishing *distributed trust*
- How can a community of individuals agree on the state of the world – or just the state of a database – without the risk of outside control or censorship
- Doing this with open-source code and cryptography turns out to be a difficult problem

Distributed Trust

- A blockchain is a decentralized, distributed and public digital ledger that is used to record transactions across **many computers** so that any involved record cannot be altered retroactively, without changing the subsequent blocks

- Distributed integrity allows the participants to **verify** and **audit** transactions independently and relatively inexpensively

Double Spend Problem

- The problem of double-spend(ing)
- This is a problem that would have to be addressed in any digital cash scheme, including schemes that preceded Bitcoin
- As with counterfeit money, double-spending leads to inflation by inflating the total amount in circulation
- This devalues the currency relative to other monetary units or goods (gold, silver) and diminishes user trust as well as the circulation and retention of the currency.
- Cryptographic techniques to prevent double-spending, while preserving transaction anonymity are blind signatures and, particularly in offline systems, secret splitting.

Which Problems Does Blockchain *Not* Solve?

- Privacy
- Throughput
- What about other properties?
  - Auditability?
  - Availability?
  - Non-repudiation?

Killer App

- So far, the killer app is cryptographic money
- Global transaction history can be found on a public ledger like Bitcoin or Ethereum
- No need for a bank or a government approving your transactions
- You can remain largely anonymous
- Transactions cannot be reverted unlike SWIFT or other government-controlled payment systems
- Don’t need intermediaries – can control your own privacy keys

Consensus Protocols

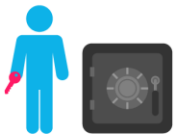
- Proof-of-Work (PoW): BTC, ETH
- Proof-of-Stake (PoS):
- Delegated Proof-of-Stake (DPoS): EOS, Tezos
- Proof-of-Authority (PoA)

POW vs. POS

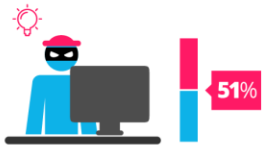
Blockgeeks

Proof of Work vs Proof of Stake

proof of work is a requirement to define an expensive computer calculation, also called mining

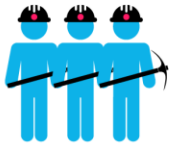


Proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake.



A reward is given to the first miner who solves each blocks problem.

The PoS system there is no block reward, so, the miners take the transaction fees.



Network miners compete to be the first to find a solution for the mathematical problem

Proof of Stake currencies can be several thousand times more cost effective.

Example: Lisk POS

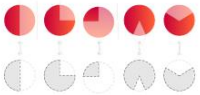
Proof of Stake

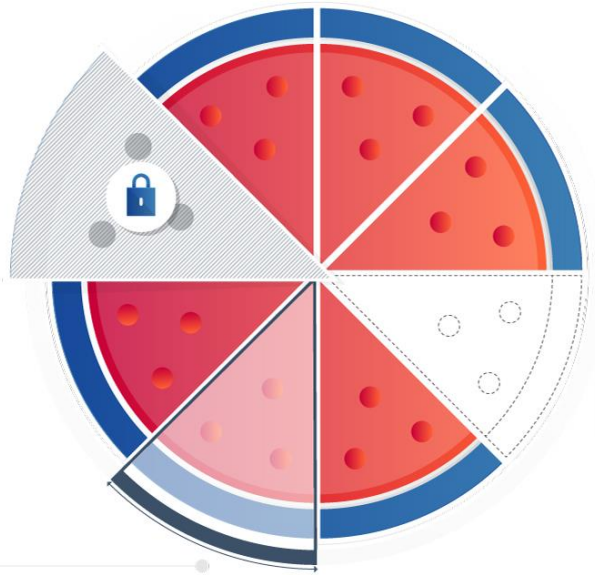
*In **Proof of Stake**, each validator owns some stake in the network, and has to lock it in order to be selected.*

1 **Anyone who holds the base cryptocurrency can become a validator,** although sometimes a locked up deposit is required.

2 A validators chance of mining a block is based **on how much of a stake (or cryptocurrency) they have.**

For example, if you owned 1% of the cryptocurrency, you would be able to mine 1% of all its transactions.





3 The PoS protocol will randomly assign the right to create a block in between selected validators, based upon the value of their stakes.

The chosen validator is rewarded by a part or the whole of the transaction fee.

Lisk ACADEMY

51% Attacks

- A double spending attack, is a potential attack against cryptocurrencies that has happened to several cryptocurrencies, e.g. due to the 51% attack.

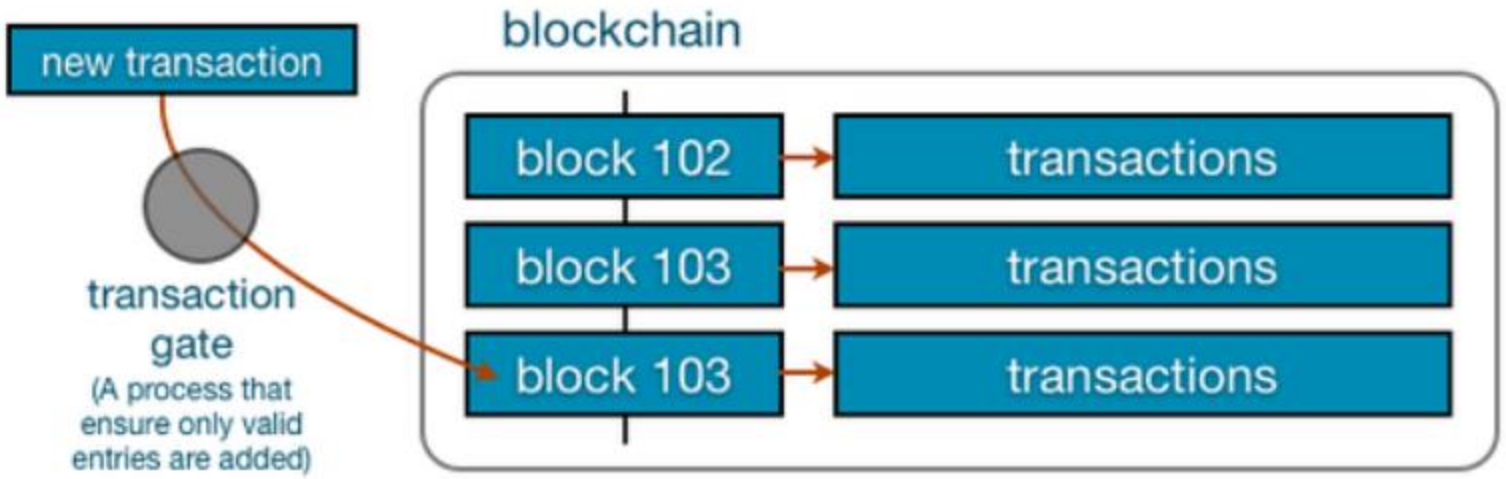
- While it hasn't happened against many of the largest cryptocurrencies, such as Bitcoin (with even the capability arising for it in 2014), it h**as happened to one of its forks**, Bitcoin Gold, then 26th largest cryptocurrency.

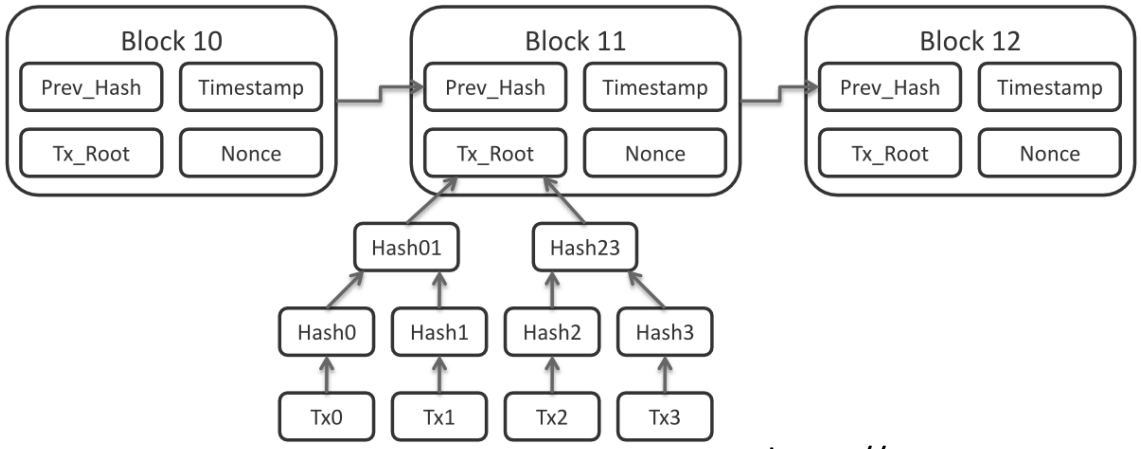
Bitcoin Gold Hack

- In 2018, Bitcoin Gold (and two other cryptocurrencies) were hit a by a successful 51% hashing attack by an unknown actor.[3] The attackers successfully committed a double spend attack on Bitcoin Gold, a cryptocurrency forked from Bitcoin in 2017.

- Approximately $18.6 million USD worth of Bitcoin Gold was transferred to a cryptocurrency exchange (typically as part of a pair transaction in exchange of a fiat currency or another cryptocurrency) and then reverted in the public ledger maintained by consensus of Proof-of-Work by exercising a >51% mine power

Blockchain Structure





https://mycryptoeconomist.com/blockchain-101/

Components of a Blockchain

Digital Ledger

- The digital ledger also known as DLT [Distributive Ledger Technology] is continually updated database of all the transactions on the blockchain. The blockchain is comprised of transactions on a block that contain all the previous blocks transaction history 'chained' together by Cryptographic science also known as Cryptography.

Consensus

- Consensus is used to verify every single transaction from all participants on the blockchain. Without combined and complete consensus on the blockchain network the transaction are not verified and therefore rejected. This keeps the integrity of the blockchain in place. Consensus is required for public blockchains and not necessarily private blockchains.

Digital Asset

- The digital asset in this case being bitcoin. The asset is the transaction item on the blockchain being transacted. This transaction item can be any number of things not only cryptocurrencies like bitcoin. There are blockchains programmed for ID information, Legal documents etc..

Network Participants

- Network participants also known as nodes on the blockchain are connected computers. These computers such yours or mine have stored the blockchain on their respective hard drives and remotely plug into it with an internet connection. This allows consensus to be made on transactions as noted above.

Hacker Makes Over $18 Million in Double-Spend Attack on Bitcoin Gold Network

Hacker made over $18 million so far

The Bitcoin Gold team says it tracked the funds stolen from exchanges to a BTG address located at GTNjvCGssb2rbLnDV1xxsHmunQdvXnY2Ft.

BTG Address

Summary		Transactions	
Address	GTNjvCGssb2rbLnDV1xxsHmunQdvXnY2Ft	No. Transactions	76
BTC Format	1AXpW4wvtjRZWsUvZ5JrSXS1sEr5ZsUaUc	Total Received	388,201.92404001 BTG
Final Balance	12,239.00 BTG	Total Send	375,962.92404001 BTG

More than 388,000 BTG coins have passed through this account, suggesting the hacker has made over $18 million from his attacks.

https://www.bleepingcomputer.com/news/security/hacker-makes-over-18-million-in-double-spend-attack-on-bitcoin-gold-network/

ZenCash 51% Attack

JUNE 3, 2018

ZenCash Statement On Double Spend Attack

English, news // double spend, exchanges, ZenCash // 17

Latest update 8 June 2018 14:01 EDT. [FAQ section added]

The Zen network was the target of a 51% attack on 2 June at approximately 8:26 pm EDT (03 June 00:26 hrs UTC). The Zen team immediately executed mitigation procedures to significantly increase the difficulty of future attacks on the network.

Sequence of events:

- 6/2 (2026 EDT) – Received warning of potential attack from one of our pool operators
- 6/2 (2034 EDT) – Immediately initiated investigation and evaluated hash power distribution
- 6/2 In parallel, contacted exchanges to increase confirmation times
- 6/2 (2042 EDT) – Investigation showed that the suspect transaction was a double spend
- 6/3 – present – In progress: Additional forensics and jointly investigating with the affected exchange
- 6/3 (0900 EDT) – Released this official announcement about the attacks(edited)
- 6/4 (1150 EDT) – Released new finding on the investigation
- 6/6 (0946 EDT) – Co-founder, Rob Viglione, issued statement responding to the attacks and dispel misconceptions

A 51% attack or double spend is a major risk for all distributed, public blockchains. All Equihash-based networks are exposed to an influx of new Equihash power and therefore the best short-term mitigation strategy is to recommend that all exchanges increase their minimum required confirmations to at least 100.

Double-Spend Observed

WHAT WE KNOW SO FAR

At the time of the attack the Zen network hash rate was 58MSol/s. It is possible that the attacker has a private mining operation large enough to conduct the attack and/or supplement with rental hash power. Net hash rate is derived from the last mined block and therefore live hash rate statistics are not available.

The suspect pool address is **`znkMXdwwxvPp9jNoSjukAbBHjCShQ8ZaLib`**. Between blocks 318165 and 318275, the attacker(s) caused multiple reorganizations of the blockchain, reverting 38 blocks in the longest reorganization. In block 318204 and 318234 the attacker(s) performed double-spend attacks.

Note: Bittrex had transaction confirmation of 150 prior to the attack and therefore was not the target of the attack.

1ST DOUBLE SPEND – 3,317.4 ZEN (NEW FINDING)

Orphaned transaction
https://explorer.zensystem.io/tx/e3a232af6d1175ad061b95f9bc12898fa22d6adcb2e9fdc9f45a2ff6711e5f93
In orphaned block
https://explorer.zensystem.io/block/000000006fbe8353edc35b8d3a08ae60f689b92a5493f59995ccd1f0209bda29
Double-spend transaction
https://explorer.zensystem.io/tx/cb072b3755547362b26fa32992380528d4f5f25b63d24bb50466a733b8edd513
Included in attacker block
https://explorer.zensystem.io/block/00000000245571c7c62059b7bf951c613c6d733242ead752767dc6e632a80128

Crypto51.app

PoW 51% Attack Cost

This is a collection of coins and the theoretical cost of a 51% attack on each network.

Learn More

Name	Symbol	Market Cap	Algorithm	Hash Rate	1h Attack Cost	NiceHash-able
Bitcoin	BTC	$67.18 B	SHA-256	42,587 PH/s	*$255,630*	1%
Ethereum	ETH	$11.54 B	Ethash	192 TH/s	*$81,959*	4%
Bitcoin Cash	BCH	$3.16 B	SHA-256	1,417 PH/s	*$8,507*	32%
Litecoin	LTC	$1.73 B	Scrypt	181 TH/s	*$16,672*	8%
Dash	DASH	$748.88 M	X11	3 PH/s	*$9,169*	20%
Ethereum Classic	ETC	$481.86 M	Ethash	10 TH/s	*$4,384*	79%
Zcash	ZEC	$358.37 M	Equihash	2 GH/s	*$17,656*	7%
Bytecoin	BCN	$143.59 M	CryptoNight	892 MH/s	*$635*	73%
Siacoin	SC	$101.57 M	Sia	848 TH/s	*$0*	0%
Electroneum	ETN	$66.68 M	CryptoNight	2 GH/s	*$1,603*	29%
Metaverse ETP	ETP	$44.32 M	Ethash	589 GH/s	$251	1,379%
MonaCoin	MONA	$41.00 M	Lyra2REv2	14 TH/s	*$523*	99%
Horizen	ZEN	$33.06 M	Equihash	167 MH/s	$1,201	108%
Bitcoin Private	BTCP	$27.60 M	Equihash	4 MH/s	$30	4,273%

How to Estimate the Costs

What is a 51% attack?

In Proof of Work (PoW) cryptocurrencies, nodes typically are set up to recognize the blockchain with the most blocks (and therefore the most hashing power) as the correct version of history. Miners with > 50% of the network hashing power can take advantage of this by sending funds to one address on the main chain, while sending the same funds to another address on a forked copy of the blockchain that they are silently mining with more hashing power than the main chain.

Since other nodes only know about the main chain, they will see the first transaction as valid, and exchanges, etc will accept this transaction as valid. This malicious node can later release these silently mined blocks, and other nodes will accept this as the new 'correct chain' since it is longer. This will cause the original transaction to effectively dissappear, and nodes will recognize the funds as being sent to the address from the new chain instead. This is known as a 'double spend' attack.

Most bigger cryptocurrencies have sufficient mining capacity behind them, making it extremely expensive to acquire the necessary hardware to pull an attack like this off. Smaller cryptocurrencies have less hashing power securing the network, making it possible to simply rent hashing power from miners on a service like Nicehash for a few hours. This significantly reduces the capital costs of an attack.

In recent weeks there have been a number of 51% attacks including a high profile attack against Bitcoin Gold a few days ago where $18 Million was stolen.

How is the attack cost calculated?

Using the prices NiceHash lists for different algorithms we are able to calculate how much it would cost to rent enough hashing power to match the current network hashing power for an hour. Nicehash does not have enough hashing power for most larger coins, so we also calculated what percentage of the needed hashing power is available from Nicehash.

Note that the attack cost does not include the block rewards that the miner will receive for mining. In some cases this can be quite significant, and reduce the attack cost by up to 80%.

Can we reduce the risk?

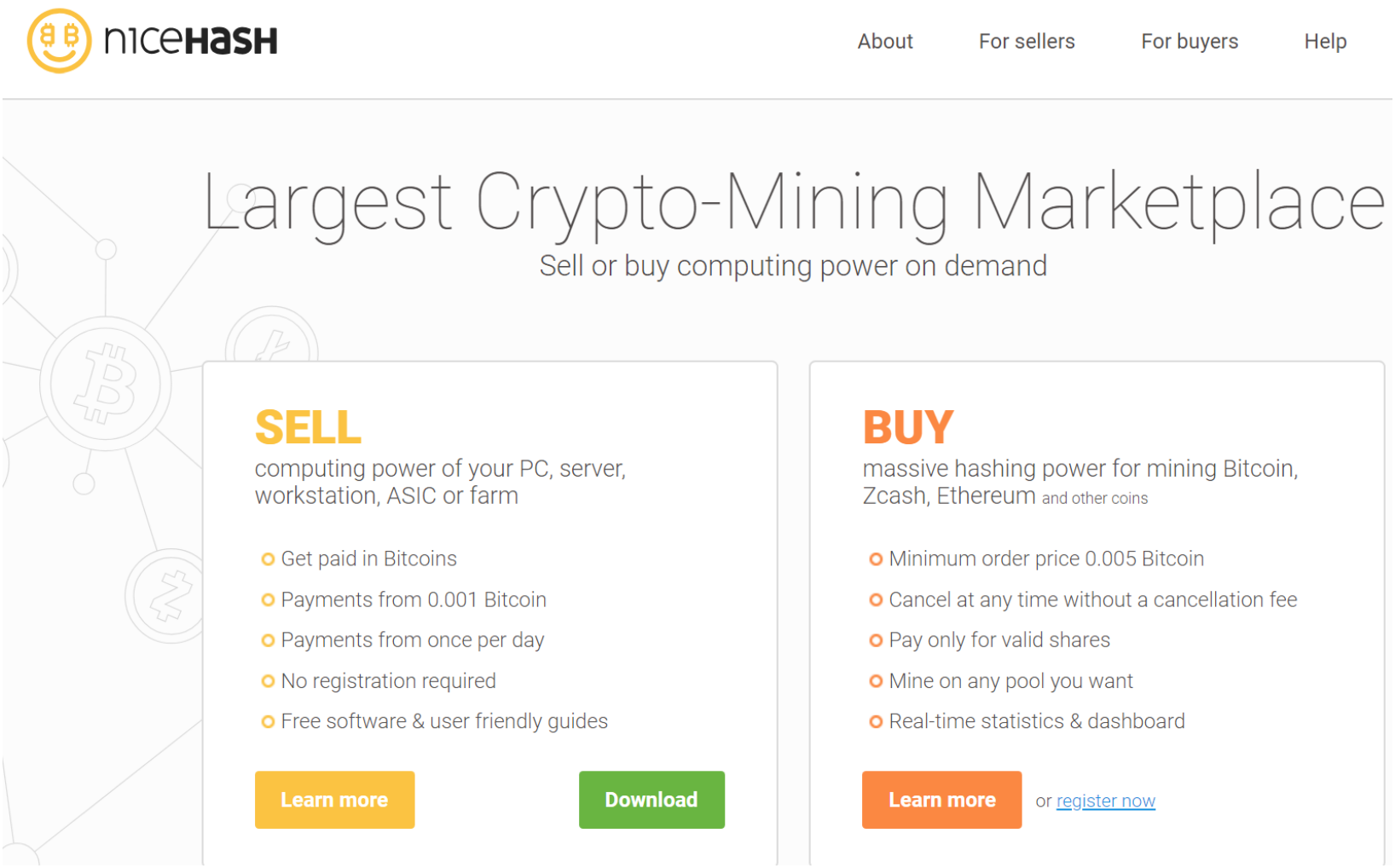
There are a number of possible solutions to this problem:

- PoS
- Coins built on top of other networks (ERC20)
- Interchain linking

Where is the data from?

Hash rates are from Mine the Coin, coin prices are from CoinMarketCap, and rental pricing is from NiceHash. The data has been spot checked for accuracy, but please let us know if any data is incorrect, and we'll do our best to fix it.

NiceHash.com



Decentralization in Bitcoin and Ethereum Networks

Decentralization in Bitcoin and Ethereum Networks

Adem Efe Gencer[1,2], Soumya Basu[1,2], Ittay Eyal[1,3], Robbert van Renesse[1,2], and Emin Gün Sirer[1,2]

[1] Initiative for Cryptocurrencies and Contracts (IC3)
[2] Computer Science Department, Cornell University
[3] Electrical Engineering Department, Technion

Abstract. Blockchain-based cryptocurrencies have demonstrated how to securely implement traditionally centralized systems, such as currencies, in a decentralized fashion. However, there have been few measurement studies on the level of decentralization they achieve in practice. We present a measurement study on various decentralization metrics of two of the leading cryptocurrencies with the largest market capitalization and user base, Bitcoin and Ethereum. We investigate the extent of decentralization by measuring the network resources of nodes and the interconnection among them, the protocol requirements affecting the operation of nodes, and the robustness of the two systems against attacks. In particular, we adapted existing internet measurement techniques and used the Falcon Relay Network as a novel measurement tool to obtain our data. We discovered that neither Bitcoin nor Ethereum has strictly better properties than the other. We also provide concrete suggestions for improving both systems.

1 Introduction

Cryptocurrencies are emerging as a new asset class, with a market capitalization of about $150B as of Sept 2017 [15], a growing ecosystem, and a diverse community. The most prominent platforms that account for over 70% of this market are Bitcoin [57] and Ethereum [28, 70]. The underlying technology, the *blockchain*, achieves consensus in a decentralized, open system and enables innovation in industries that conventionally relied upon trusted authorities. Some examples of such services include land record management [3], domain name registration [51], and voting [55]. The key feature that empowers such services and makes these platforms interesting is *decentralization*. Without it, such services are technologically easy to construct but require trust in a centralized administrator.

Decentralization is a property regarding the fragmentation of control over the protocol. In the Bitcoin and Ethereum protocols, users submit transactions for miners to sequence into blocks. Better decentralization of miners means higher resistance against censorship of individual transactions. For communication, Bitcoin and Ethereum also have a peer-to-peer network for disseminating block and transaction information. Both Bitcoin and Ethereum also contain *full nodes*,

Mining on cryptocurrency networks is a complex process that typically requires large computation power. With the current mining difficulty of Bitcoin and Ethereum, using commodity hardware to generate blocks is not feasible, which centralizes the mining process somewhat. However, as long as there are many different entities mining, the system is still decentralized. We compare the decentralization of the mining process between Bitcoin and Ethereum.

Distribution of Mining Power in Bitcoin and Ethereum Networks

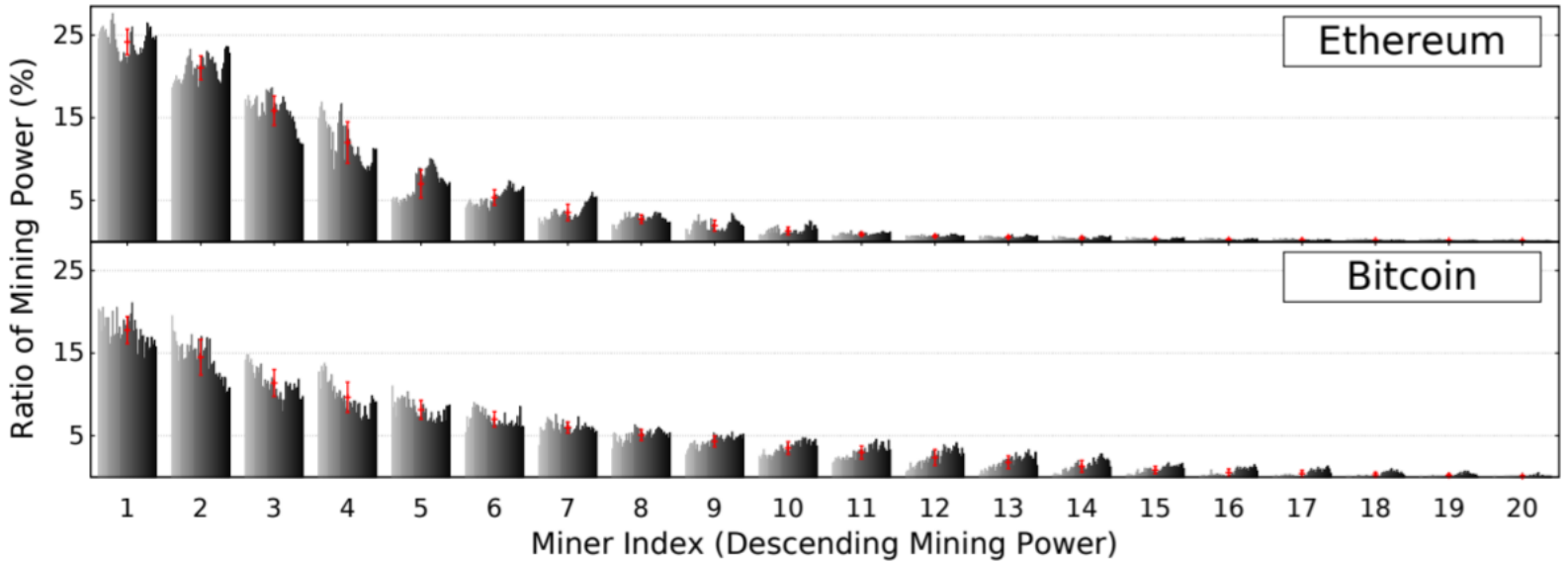


Fig. 4: Distribution of mining power in Bitcoin and Ethereum networks. Bars indicate observed standard deviation from the average.

Consolidation Effects

- Figure 4 illustrates that, in Bitcoin, the weekly mining power of a single entity has never exceeded 21% of the overall power. In contrast, the top Ethereum miner has never had less than 21% of the mining power. Moreover, the top four Bitcoin miners have more than 53% of the average mining power. On average, 61% of the weekly power was shared by only three Ethereum miners. These observations suggest a slightly more centralized mining process in Ethereum

Really Decentralized?

- Even 90% of the mining power seems to be controlled by only 16 miners in Bitcoin and only 11 mine
- Results show that a Byzantine quorum system [53] of size 20 could achieve better decentralization than proof-of-work mining at a much lower resource cost.
- This shows that further research is necessary to create a permissionless consensus protocol without such a high degree of centralization.

Attack Possibilities

- The argument that mining pools provide a degree of decentralization due to mining pool participants having a check on pool operator behavior has no empirical support. For instance, censorship attacks by pool operators are are difficult, if not impossible, to detect by pool participants.

- Additionally, when miners exceeded the 51% threshold on **three separate occasions** in Bitcoin's history, the pool participants did not disband the pool despite clear evidence of a behaviour widely understood to be unacceptable.

- Most crucially, whether mining pools provide a degree of decentralization is inconsequential for the purposes of this paper, which provides an accurate historical account. We report what happened at the time the blocks were mined, as recorded on the blockchain. As such, it is immaterial whether the miners were part of a pool or whether they were solo miners. At the time a block was committed to the chain, pool participants were plaintively cooperating as part of the same mining entity.

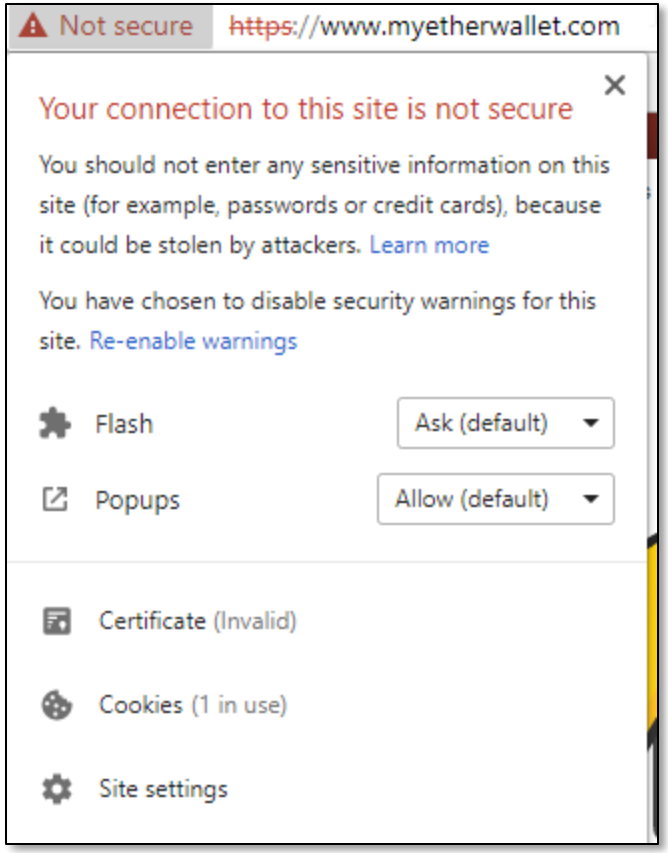
MyEtherWallet DNS Hack

MyEtherWallet Warns That A "Couple" Of Its DNS Servers Have Been Hacked

Update: Data from EtherScan shows that over $150k worth of ETH has been stolen in the DNS hack. Starting from 07:17 this morning, 179 inbound transactions totaling 216.06 ETH were sent to ETH address 0x1d50588C0aa11959A5c28831ce3DC5F1D3120d29. At 10:15, the attacker sent 215 ETH to 0x68ca85dbf8eba69fb70ecdb78e0895f7cd94da83.

One MEW user on Reddit explained how they lost 0.9 ETH when their connection was intercepted as they logged in:

> *Woke up today, Put my computer on, went on to myetherwallet and saw that myetherwallet had a invalid connection certificate in the corner. I thought this was odd. https://i.imgur.com/2x9d7bR.png . So I double checked the url address, triple checked it, went on google, got the url . Used EAL to confirm it wasn't a phishing site. And even though every part of my body told me not to try and log in, I did. As soon as I logged in, there was a countdown for about 10 seconds and A tx was made sending the available money I had on the wallet to another wallet, "0x1d50588C0aa11959A5c28831ce3DC5F1D3120d29."*



https://cointelegraph.com/news/myetherwallet-warns-that-a-couple-of-its-dns-servers-have-been-hacked

Hardware Wallets

- Private keys are never exposed to your computer.
- The hardware is immune to computer viruses.
- Your hardware requires you to confirm a transaction on your device (not the app on your computer) before any coins can be spent.
- Most hardwares are encrypted with pin #'s, like your debit card, which adds another layer security.
- The hardware company's software is usually open source which allows users to validate the entire operation of the device.
- Hardware wallets can host multiple cryptocurrencies.

Researcher Demonstrates how Vulnerable Ledger Nano S Wallets are to Hacking (March 2018)

- Weeks after the company confirmed a flaw in its wallets which makes them susceptible to man-in-the-middle-attacks, independent security researcher Saleem Rashid has demonstrated a new attack vector hackers can employ to break your Ledger Nano S and steal your precious coins – both physically and remotely.

- "The vulnerability arose due to Ledger's use of a custom architecture to work around many of the limitations of their Secure Element," Rashid explains in a blog post. "An attacker can exploit this vulnerability to compromise the device before the user receives it, or to steal private keys from the device physically or, in some scenarios, remotely."

- The researcher has outlined at least three separate attack vectors, but his report focuses on the case of "supply chain attacks" which do not require infecting target computers with additional malware, nor do they insist on the user to confirm any transactions.

https://thenextweb.com/hardfork/2018/03/20/ledger-nano-s-hack-cryptocurrency/

Breaking the Ledger Security Model

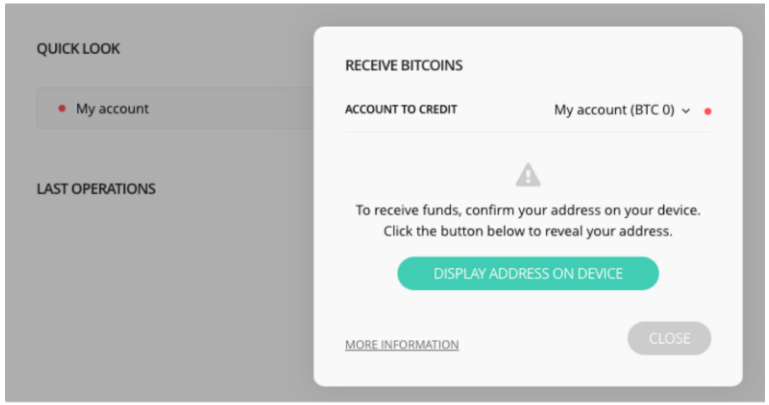
- Physical access before setup of the seed
  - Also known as a "supply chain attack", this is the focus of this article. It does not require malware on the target computer, nor does it require the user to confirm any transactions. Despite claims otherwise, I have demonstrated this attack on a real Ledger Nano S. Furthermore, I sent the source code to Ledger a few months ago, so they could reproduce it
- Physical access after setup
  - This is commonly known as an "Evil Maid attack". This attack would allow you to extract the PIN, recovery seed and any BIP-39 passphrases used, provided the device is used at least once after you attack it.
  - As before, this does not require malware on the computer, nor does it require the user to confirm any transactions. It simply requires an attacker to install a custom MCU firmware that can exfiltrate the private keys without the user's knowledge, next time they use it.
- Malware (with a hint of social engineering)
  - This attack would require the user to update the MCU firmware on an infected computer. This could be achieved by displaying an error message that asks the user to reconnect the device with the left button held down (to enter the MCU bootloader). Then the malware can update the MCU with malicious code, **allowing the malware to take control of the trusted display and confirmation buttons** on the device.

https://saleemrashid.com/2018/03/20/breaking-ledger-security-model/

MATM Against Ledger Wallet Software

Ledger

WHO WE ARE WHAT WE DO LATEST NI



Man in the Middle Attack – Am I at risk?

05/02/2018 | BLOG POSTS, SECURITY

The website bitcoin.com published a blog post on Saturday, February 3rd, titled Ledger Addresses Man in the Middle Attack That Threatens Millions of Hardware Wallets. Some of the claims made in this post are unfortunately incorrect.

This is not a Ledger security flaw. Ledger users are not at risk, as long as they verify their new receive address on their device when they share it to receive fund. As far as we know, no user has ever lost any coins because of what remains a proof of concept.

Some Background

We initially designed Ledger hardware wallets because computers cannot be considered secure. A malware or virus could replace the receiving address on a computer with another one, tricking the user into sending funds to an unintended third-party (possibly the attacker).

Hardware wallets provide an isolation layer between the computer and the seed (your private keys). However, users must always ensure that they are sending coins to the correct address when transacting.

https://www.ledger.fr/2018/02/05/man-middle-attack-risk/

Second Factor to the Rescue

The Proof of Concept attack

Researchers published a proof of concept attack in which a malware modifies the Ledger Chrome application in order to edit the received address displayed on the computer screen.

As far as we know, this is only a proof of concept phishing attack and no Ledger user has ever been fooled using this technique. We were already aware of this scenario: computers cannot be considered secure, and therefore you cannot trust what you see on the screen. That's the very reason why we decided to create the Ledger hardware wallet in the first place.

We would like to insist on the fact that in a threat model where the attacker is able to do anything on the computer, it is impossible to trust what is displayed on the computer screen. The only thing users can completely trust is what is displayed on the screen of their Ledger hardware wallet. The Ledger Wallet Bitcoin Chrome application also has a dedicated icon (third one from the left hand side, see image above) allowing the user to display the receiving address on their Ledger device. When the user clicks on this icon, the correct address is generated by the wallet and displayed on the Ledger hardware wallet's screen. **This is the only information you can trust.**

Action Points

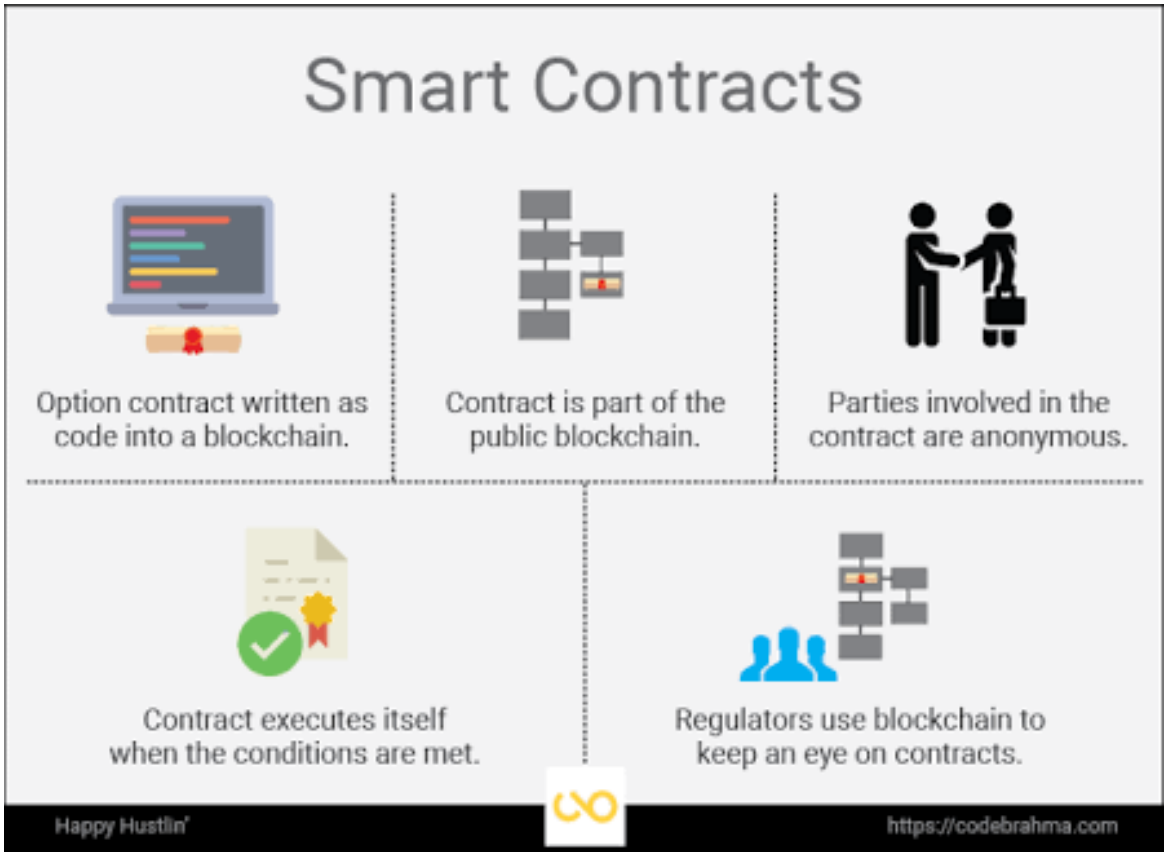
At Ledger, we strive to provide our users with an easy and secure way to manage their crypto assets. In order to avoid any misuse, we will keep providing our community with additional services and information, starting with the ones listed below.

- Software update: we released an update to the Ledger Wallet Bitcoin Chrome application that will request users to verify destination addresses on their Ledger hardware device – not just on the screen of their computer. Bitcoin & altcoins are getting the new feature (ETH and XRP apps will benefit from the feature in the new global release)

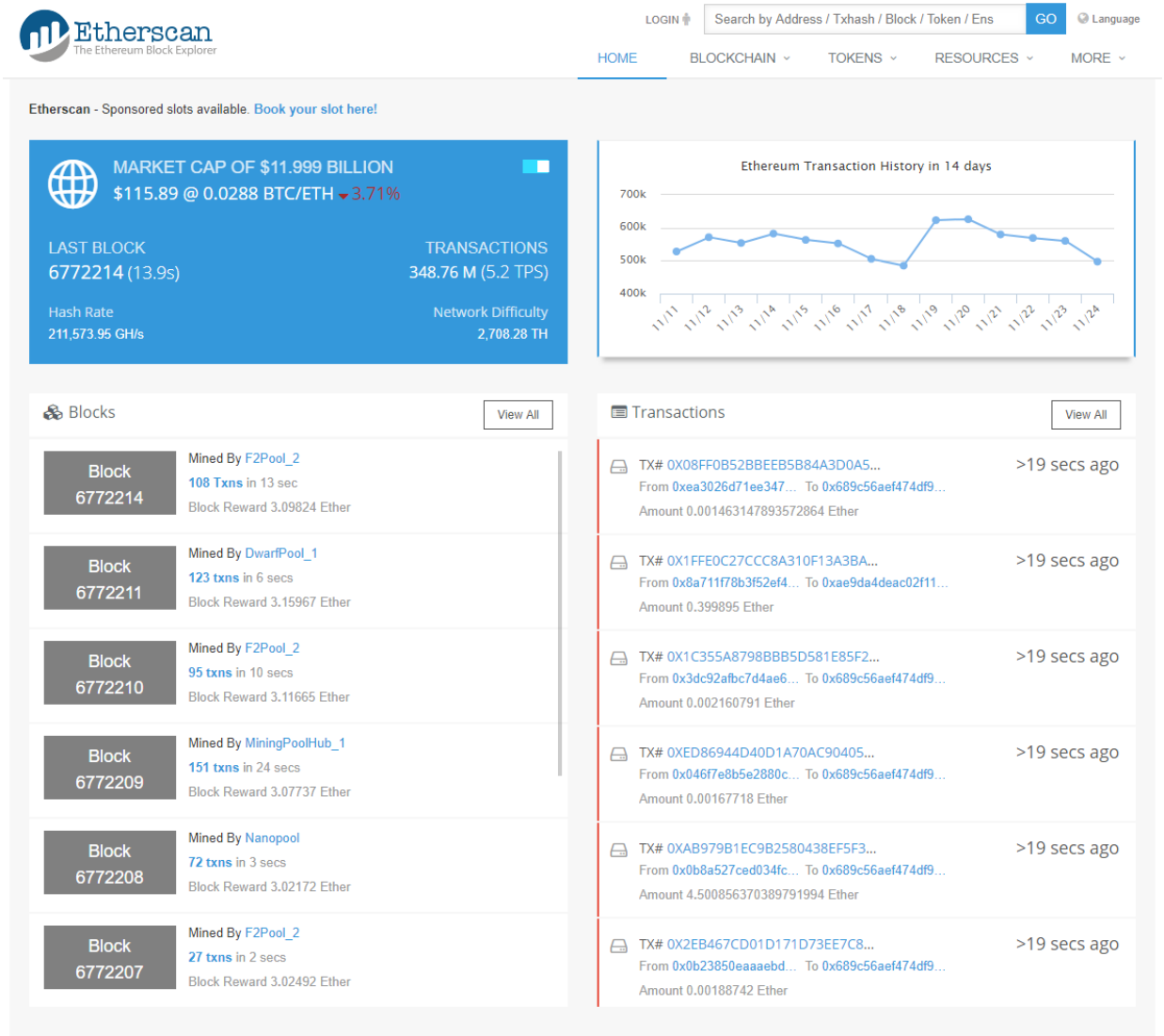
February 2018:

Software update: we released an update to the Ledger Wallet Bitcoin Chrome application that will request users to verify destination addresses on their Ledger hardware device – not just on the screen of their computer. Bitcoin & altcoins are getting the new feature (ETH and XRP apps will benefit from the feature in the new global release)

Smart Contracts



https://etherscan.io



Solidity Code


```
1  contract MetaCoin {
2    mapping (address => uint) balances;
3
4    function MetaCoin() {
5      balances[tx.origin] = 10000;
6    }
7
8    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
9      if (balances[msg.sender] < amount) return false;
10     balances[msg.sender] -= amount;
11     balances[receiver] += amount;
12     return true;
13   }
14
15   function getBalance(address addr) returns(uint) {
16     return balances[addr];
17   }
18 }
19
```

EVM

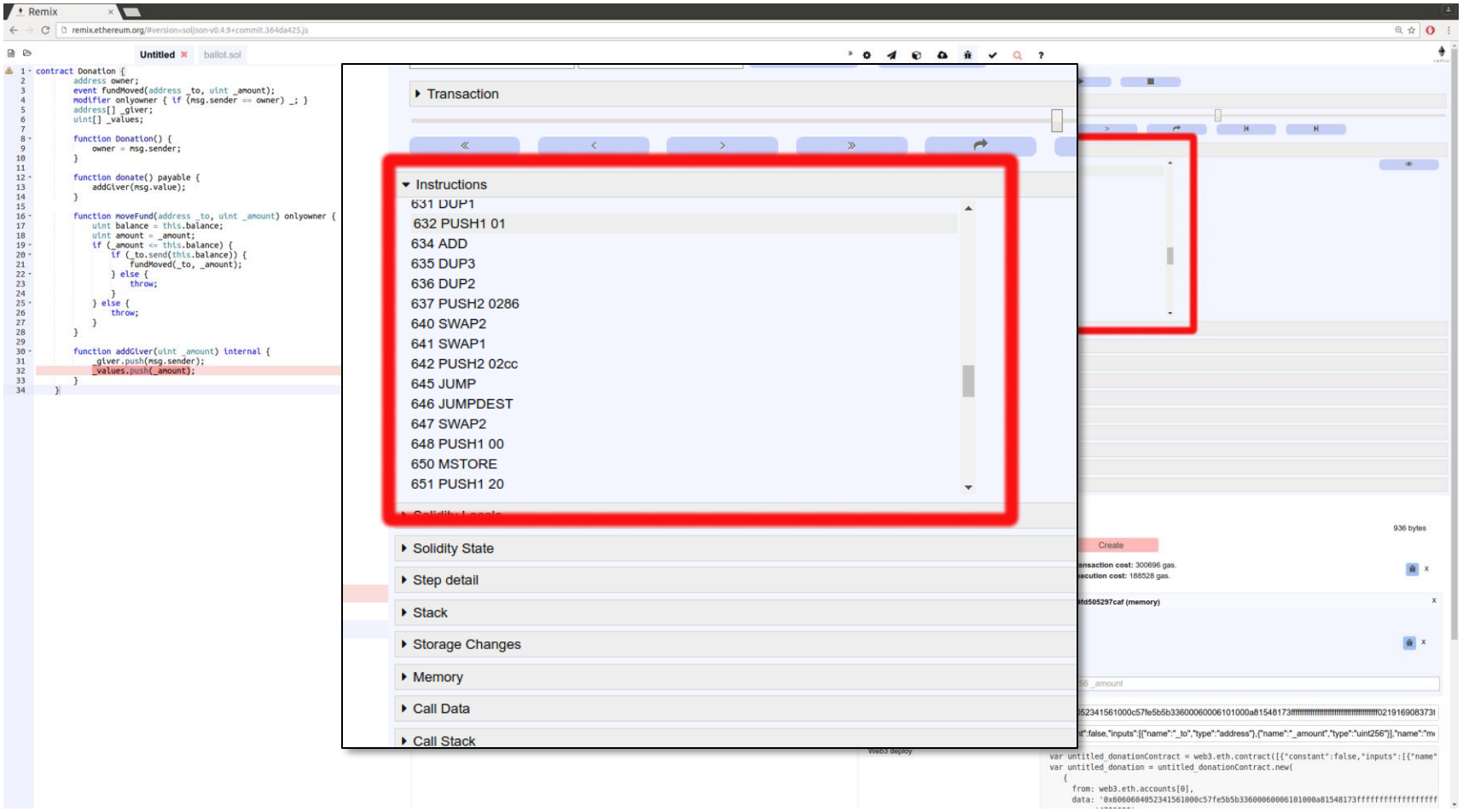
- Ethereum is a decentralized virtual machine, which runs programs — called contracts — upon request of users. Contracts are written in a Turing-complete bytecode language, called EVM bytecode. Roughly, a contract is a set of functions, each one defined by a sequence of bytecode instructions. A remarkable feature of contracts is that they can transfer ether (a cryptocurrency similar to Bitcoin) to/from users and to other contracts.

- Users send transactions to the Ethereum network in order to: (i) create new contracts; (ii) invoke functions of a contract; (iii) transfer ether to contracts or to other users. All the transactions are recorded on a public, append-only data structure, called blockchain. The sequence of transactions on the blockchain determines the state of each contract, and the balance of each user.

Execution Fees

- Each function invocation is ideally executed by all miners in the Ethereum network. Miners are incentivized to do such work by the execution fees paid by the users which invoke functions.
- Besides being used as incentives, execution fees also protect against denial-of-service attacks, where an adversary tries to slow down the network by requesting time-consuming computations.
- Execution fees are defined in terms of gas and gas price, and their product represents the cost paid by the user to execute code. More specifically, the transaction which triggers the invocation specifies the gas limit up to which the user is willing to pay, and the price per unit of gas.
- Roughly, the higher is the price per unit, the higher is the chance that miners will choose to execute the transaction. Each EVM operation consumes a certain amount of gas, and the overall fee depends on the whole sequence of operations executed by miners.

EVM Instruction Stream in Remix



https://remix.readthedocs.io/en/latest/tutorial_debug.html

Decentralized Execution

- Since contracts have an economic value, it is crucial to guarantee that their execution is performed correctly. To this purpose, Ethereum does not rely on a trusted central authority: rather, each transaction is processed by a large network of mutually untrusted peers — called miners. Potential conflicts in the execution of contracts (due e.g., to failures or attacks) are resolved through a consensus protocol based on "proof-of-work" puzzles. Ideally, the execution of contracts is correct whenever the adversary does not control the majority of the computational power of the network.

- The security of the consensus protocol relies on the assumption that honest miners are rational, i.e. that it is more convenient for a miner to follow the protocol than to try to attack it. To make this assumption hold, miners receive some economic incentives for performing the (time-consuming) computations required by the protocol. Part of these incentives is given by the execution fees paid by users upon each transaction. These fees bound the execution steps of a transaction, so preventing from denial-of-service attacks where users try to overwhelm the network with time-consuming computations

A survey of Attacks on Ethereum Smart Contracts

A survey of attacks on Ethereum smart contracts

Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli

Università degli Studi di Cagliari, Cagliari, Italy
{atzeinicola,bart,t.cimoli}@unica.it

Abstract. Smart contracts are computer programs that can be correctly executed by a network of mutually distrusting nodes, without the need of an external trusted authority. Since smart contracts handle and transfer assets of considerable value, besides their correct execution it is also crucial that their implementation is secure against attacks which aim at stealing or tampering the assets. We study this problem in Ethereum, the most well-known and used framework for smart contracts so far. We analyse the security vulnerabilities of Ethereum smart contracts, providing a taxonomy of common programming pitfalls which may lead to vulnerabilities. We show a series of attacks which exploit these vulnerabilities, allowing an adversary to steal money or cause other damage.

1 Introduction

The success of Bitcoin, a decentralised cryptographic currency that reached a capitalisation of 10 billions of dollars since its launch in 2009, has raised considerable interest both in industry and in academia. Industries — as well as national governments [48,55] — are attracted by the "disruptive" potential of the *blockchain*, the underlying technology of cryptocurrencies. Basically, a blockchain is an append-only data structure maintained by the nodes of a peer-to-peer network. Cryptocurrencies use the blockchain as a public ledger where they record all the transfers of currency, in order to avoid double-spending of money.

Although Bitcoin is the most paradigmatic application of blockchain technologies, there are other applications far beyond cryptocurrencies: e.g., financial products and services, tracking the ownership of various kinds of properties, digital identity verification, voting, *etc.* A hot topic is how to leverage on blockchain technologies to implement *smart contracts* [34,54]. Very abstractly, smart contracts are agreements between mutually distrusting participants, which are automatically enforced by the consensus mechanism of the blockchain — without relying on a trusted authority.

The most prominent framework for smart contracts is Ethereum [32], whose capitalisation has reached 1 billion dollars since its launch in July 2015[1]. In Ethereum, smart contracts are rendered as computer programs, written in a Turing-complete language. The consensus protocol of Ethereum, which specifies how the nodes of the peer-to-peer network extend the blockchain, has the goal

[1] https://coinmarketcap.com/currencies/ethereum

```
1  contract AWallet{
2      address owner;
3      mapping (address => uint) public outflow;
4
5      function AWallet(){ owner = msg.sender; }
6
7      function pay(uint amount, address recipient) returns (bool){
8          if (msg.sender != owner ||  msg.value != 0) throw;
9          if (amount > this.balance) return false;
10         outflow[recipient] += amount;
11         if (!recipient.send(amount)) throw;
12         return true;
13     }
14 }
```

Taxonomy of Vulnerabilities

Level	Cause of vulnerability	Attacks
Solidity	Call to the unknown	4.1
	Gasless send	4.2
	Exception disorders	4.2, 4.5
	Type casts	—
	Reentrancy	4.1
	Keeping secrets	4.3
EVM	Immutable bugs	4.4, 4.5
	Ether lost in trasfer	—
	Stack size limit	4.5
Blockchain	Unpredictable state	4.5, 4.6
	Generating randomness	—
	Time constraints	4.5

Table 1. Taxonomy of vulnerabilities in Ethereum smart contracts.

Call to the unknown

- Some of the primitives used in Solidity to invoke functions and to transfer ether may have the side effect of invoking the fallback function of the callee/recipient.

  – `call` invokes a function (of another contract, or of itself), and transfers ether to the callee. E.g., one can invoke the function `ping` of contract `c` as follows:

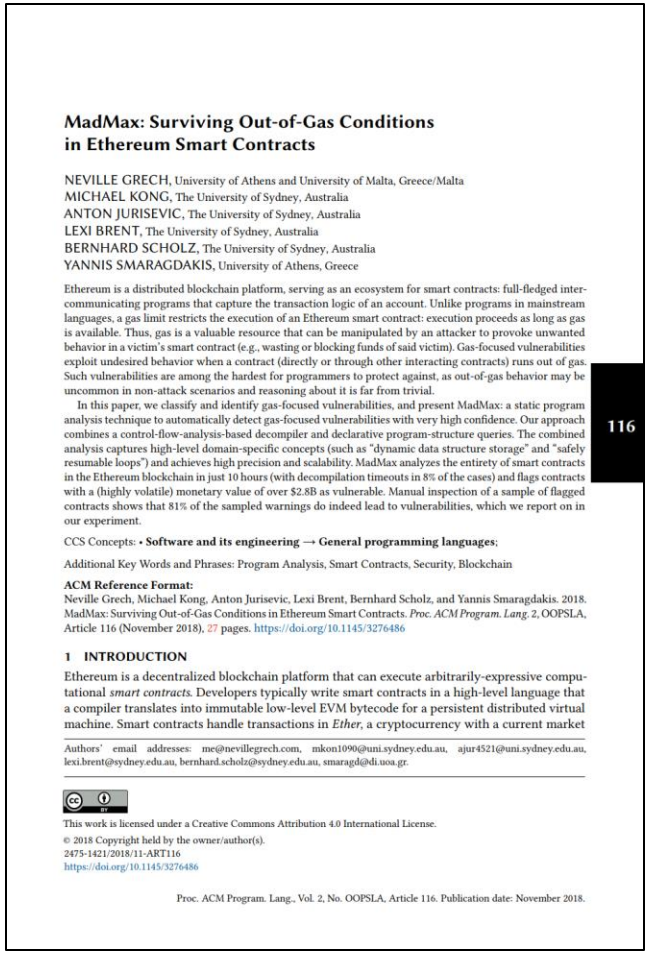
  ```
  c.call.value(amount)(bytes4(sha3("ping(uint256)")),n);
  ```

  where the called function is identified by the first 4 bytes of its hashed signature, `amount` determines how many *wei* have to be transferred to `c`, and `n` is the actual parameter of `ping`. Remarkably, if a function with the given signature does not exist at address `c`, then the fallback function of `c` is executed, instead[8].

Gasless send

- When using the function send to transfer ether to a contract, it is possible to incur in an out-of-gas exception. This may be quite unexpected by programmers, because transferring ether is not generally associated to executing code. The reason behind this exception is subtle.
- First, note that c.send(amount) is compiled in the same way of a call with empty signature, but the actual number of gas units available to the callee is always bound by 230011. Now, since the call has no signature, it will invoke the callee's fallback function.
- However, 2300 units of gas only allow to execute a limited set of bytecode instructions, e.g. those which do not alter the state of the contract. In any other case, the call will end up in an out-of-gas exception.

Gas budgeting



MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts

NEVILLE GRECH, University of Athens and University of Malta, Greece/Malta
MICHAEL KONG, The University of Sydney, Australia
ANTON JURISEVIC, The University of Sydney, Australia
LEXI BRENT, The University of Sydney, Australia
BERNHARD SCHOLZ, The University of Sydney, Australia
YANNIS SMARAGDAKIS, University of Athens, Greece

Ethereum is a distributed blockchain platform, serving as an ecosystem for smart contracts: full-fledged inter-communicating programs that capture the transaction logic of an account. Unlike programs in mainstream languages, a gas limit restricts the execution of an Ethereum smart contract: execution proceeds as long as gas is available. Thus, gas is a valuable resource that can be manipulated by an attacker to provoke unwanted behavior in a victim's smart contract (e.g., wasting or blocking funds of said victim). Gas-focused vulnerabilities exploit undesired behavior when a contract (directly or through other interacting contracts) runs out of gas. Such vulnerabilities are among the hardest for programmers to protect against, as out-of-gas behavior may be uncommon in non-attack scenarios and reasoning about it is far from trivial.

In this paper, we classify and identify gas-focused vulnerabilities, and present MadMax: a static program analysis technique to automatically detect gas-focused vulnerabilities with very high confidence. Our approach combines a control-flow-analysis-based decompiler and declarative program-structure queries. The combined analysis captures high-level domain-specific concepts (such as "dynamic data structure storage" and "safely resumable loops") and achieves high precision and scalability. MadMax analyzes the entirety of smart contracts in the Ethereum blockchain in just 10 hours (with decompilation timeouts in 8% of the cases) and flags contracts with a (highly volatile) monetary value of over $2.8B as vulnerable. Manual inspection of a sample of flagged contracts shows that 81% of the sampled warnings do indeed lead to vulnerabilities, which we report on in our experiment.

116

CCS Concepts: • **Software and its engineering → General programming languages**;

Additional Key Words and Phrases: Program Analysis, Smart Contracts, Security, Blockchain

ACM Reference Format:
Neville Grech, Michael Kong, Anton Jurisevic, Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis. 2018. MadMax: Surviving Out-of-Gas Conditions in Ethereum Smart Contracts. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 116 (November 2018), 27 pages. https://doi.org/10.1145/3276486

1 INTRODUCTION

Ethereum is a decentralized blockchain platform that can execute arbitrarily-expressive computational *smart contracts*. Developers typically write smart contracts in a high-level language that a compiler translates into immutable low-level EVM bytecode for a persistent distributed virtual machine. Smart contracts handle transactions in *Ether*, a cryptocurrency with a current market

Authors' email addresses: me@nevillegrech.com, mkon1090@uni.sydney.edu.au, ajur4521@uni.sydney.edu.au, lexi.brent@sydney.edu.au, bernhard.scholz@sydney.edu.au, smaragd@di.uoa.gr.

2475-1421/2018/11-ART116
https://doi.org/10.1145/3276486

In this work, we present MadMax: a static program analysis framework for detecting gas-focused vulnerabilities in smart contracts. MadMax is a static analysis pipeline consisting of a decompiler (from low-level EVM bytecode to a structured intermediate language) and a logic-based analysis specification producing a high-level program model. MadMax is highly efficient and effective: it analyzes the whole Ethereum blockchain in 10 hours and reports numerous vulnerable contracts holding a total value exceeding $2.8B, with high precision, as determined from a random sample.

Naïve loops, an example


```
contract NaiveBank {
  struct Account {
    address addr;
    uint balance;
  }
  Account accounts[];

  function applyInterest() returns (uint) {
    for (uint i = 0; i < accounts.length; i++) {
      // apply 5 percent interest
      accounts[i].balance = accounts[i].balance * 105 / 100;
    }
    return accounts.length;
  }
}
```

Non-Isolated External Calls (Wallet Griefing)

- Non-Isolated External Calls (Wallet Griefing) In addition to running out of gas because of unbounded, externally-controlled data structures, a contract may run into trouble because of invoking external functionality that may itself throw an out-of-gas exception.
- This is not a realistic threat in a direct setting: an external call to an unknown party is by definition untrusted, and therefore the contract programmer is highly likely to have considered malicious behavious

Calling pay

We illustrate the behaviour of send through a small example, involving a contract C who sends ether through function pay, and two recipients D1, D2

```
1  contract C {
2    function pay(uint n, address d){
3      d.send(n);
4    }
5  }
```

```
6  contract D1 {
7      uint public count = 0;
8      function() { count++; }
9  }
10 contract D2 { function() {} }
```

There are three possible cases to execute `pay`:

- `n` $\neq 0$ and `d` $=$ `D1`. The `send` in `C` fails with an out-of-gas exception, because 2300 units of gas are not enough to execute the state-updating `D1`'s fallback.
- `n` $\neq 0$ and `d` $=$ `D2`. The `send` in `C` succeeds, because 2300 units of gas are enough to execute the empty fallback of `D2`.
- `n` $= 0$ and `d` $\in \{$`D1`, `D2`$\}$. For compiler versions $< 0.4.0$, the `send` in `C` fails with an out-of-gas exception, since the gas is not enough to execute any fallback, not even an empty one. For compiler versions $\geq 0.4.0$, the behaviour is the same as in one of the previous two cases, according whether `d` $=$ `D1` or `d` $=$ `D2`.

Type casts

- The Solidity compiler can detects some type errors (e.g., assigning an integer value to a variable of type string). Types are also used in direct calls: the caller must declare the callee's interface, and cast to it the callee's address when performing the call.

```
contract Alice { function ping(uint) returns (uint) }
contract Bob   { function pong(Alice c){ c.ping(42); } }
```

The signature of `pong` informs the compiler that `c` adheres to interface `Alice`. However, the compiler only checks whether the interface declares the function `ping`, while it does *not* check that: (i) `c` is the address of contract `Alice`; (ii) the interface declared by `Bob` matches `Alice`'s actual interface. A similar situation happens with explicit type casts, e.g. `Alice(c).ping()`, where `c` is an address.

The DAO

- The DAO was a complex Smart Contract with a focus on fair, decentralized operations. In order to allow investors to leave the organization in the case of a disagreement,
- The DAO was created with an exit or a ‘split function’. This function allowed users to revert the involvement process and to have the Ether they had sent to The DAO returned.
- If someone wanted to leave The DAO, they would create their own Child DAOs, wait 28 days and then approve their proposal to send Ether to another address.

https://coincodex.com/article/50/the-dao-hack-what-happened-and-what-followed/

The Hack

- On June 18, it was noticed that funds were leaving The DAO and the Ether balance of the smart contract was being drained. Around 3.6M Ether worth approximately $70M were drained by a hacker in a few hours.
- The hacker was able to get the DAO smart contract to return Ether multiple times before it could update its own balance.
  - There were two main flaws that allowed this to take place, firstly the smart contract sent the Ether and then updated the internal token balance.
  - Secondly, The DAO coders had also failed to consider the possibility of a recursive call that could act in such a way.
- The hack resulted in the proposal of a soft fork that would stop the stolen funds from being spent, however, this never took place after a bug was discovered within the implementation protocol. This opened up the possibility of a hard fork with wider reaching implications.

The Hard Fork

- A hard fork was proposed that would return all the Ether stolen The DAO in the form of a refund smart contract. The new contract could only withdraw and investors in The DAO could make refund requests for lost Ether.
- While it makes perfect sense to seek to reimburse the victims of the attack, the hard fork uncovered a number of arguments that are still prevalent in the world of cryptocurrency today.
  - Some opposed the hard fork and argued that the original statement of The DAO terms and conditions could never be changed.
  - They also felt that the blockchain should be free from censorship and things that take place on the blockchain shouldn't be changed even in the event of negative outcomes.
  - Opponents of these arguments felt that the hacker could not be allowed to profit from his actions and that returning the funds would keep blockchain projects free from regulation and litigation.
- The hard fork also made sense as it only returned funds to the original investors and would also help to stabilize the price of Ether.

What Happened: July 20 2016

- The final decision was voted on and approved by Ether holders, with **89% voting** for the hard fork and as a result, it took place on July 20 2016 during the 1920000th block.
- The immediate result of this was the creation of Ethereum Classic ETC, 7.22% which shares all the data on the Ethereum blockchain up until block 1920000.
- The creation of Ethereum Classic showed that hard forks were **very much possible** and it can be said that the creation of the second Ethereum currency has had an influence on the creators of subsequent Bitcoin BTC, 7.72% forks.
- It also became clear that while the DAO was great idea, it was not implemented correctly and in order to move forward successfully blockchain projects would have to implement rigid security protocols.

The DAO Hack

The attacker was analyzing DAO.sol, and noticed that the 'splitDAO' function was vulnerable to the recursive send pattern we've described above: this function updates user balances and totals at the end, so if we can get any of the function calls before this happens to call splitDAO again, we get the **infinite recursion** that can be used to move as many funds as we want (code comments are marked with XXXXX, you may have to scroll to see them)...

http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/

Call Split More than Once

The basic idea is this: propose a split. Execute the split. When the DAO goes to withdraw your reward, call the function to execute a split before that withdrawal finishes.

```
function splitDAO(
  uint _proposalID,
  address _newCurator
) noEther onlyTokenholders returns (bool _success) {

  ...
  // XXXXX Move ether and assign new Tokens.  Notice how this is done first!
  uint fundsToBeMoved =
      (balances[msg.sender] * p.splitData[0].splitBalance) /
      p.splitData[0].totalSupply;
  if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false
) // XXXXX This is the line the attacker wants to run more than once
      throw;

  ...
  // Burn DAO Tokens
  Transfer(msg.sender, 0, balances[msg.sender]);
  withdrawRewardFor(msg.sender); // be nice, and get his rewards
  // XXXXX Notice the preceding line is critically before the next few
  totalSupply -= balances[msg.sender]; // XXXXX AND THIS IS DONE LAST
  balances[msg.sender] = 0; // XXXXX AND THIS IS DONE LAST TOO
  paidOut[msg.sender] = 0;
  return true;
}
```

Moving Funds Multiple Times

- Basically the attacker is using this to transfer more tokens than they should be able to into their child DAO.
- How does the DAO decide how many tokens to move? Using the balances array of course:
- Because p.splitData[0] is going to be the same every time the attacker calls this function (it's a property of the proposal p, not the general state of the DAO), and because the attacker can call this function from withdrawRewardFor **before the balances array is updated**, the attacker can get this code to run arbitrarily many times using the described attack, with fundsToBeMoved coming out to the same value each time.

```
uint fundsToBeMoved = (balances[msg.sender] * p.splitData[0].splitBalance) / p.splitDat
a[0].totalSupply;
```

Recursive Call in MakerDAO

- Our team is blessed to have Dr. Christian Reitwießner, Father of Solidity, as its Advisor. During the early development of the DAO Framework 1.1 and thanks to his guidance we were made aware of a generic vulnerability common to all Ethereum smart contracts. We promptly circumvented this so-called “recursive call vulnerability” or “race to empty” from the DAO Framework 1.1 as can be seen on line 580:

```
// we are setting this here before the CALL() value transfer to
// assure that in the case of a malicious recipient contract trying
// to call executeProposal() recursively money can't be transferred
// multiple times out of the DAO
p.proposalPassed = true;
```

More DAOs

- Three days ago this design vulnerability potential was raised in a blog post which subsequently led to the discovery of such an issue in an unrelated project, MakerDAO. This was highlighted in a reddit post, with MakerDAO being able to drain their own funds safely before the vulnerability could be exploited.

- Around 12 hours ago user Eththrowa on the DAOHub Forum spotted that while we had identified the vulnerability in one aspect of the DAO Framework, the existing (and deployed) DAO reward account mechanism was affected. His message and our prompt confirmation can be found here.

- We issued a fix immediately as part of the DAO Framework 1.1 milestone.

https://blog.slock.it/no-dao-funds-at-risk-following-the-ethereum-smart-contract-recursive-call-bug-discovery-29f482d348b