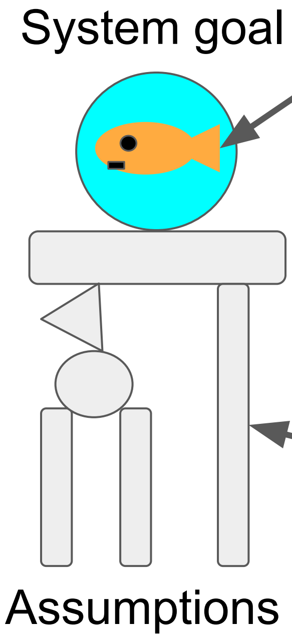
MIT Digital Currency Initiative and the University of Brasilia present

Cryptocurrency Design and Engineering

Lecture 18: Security – What can go wrong?
Taught by: Ethan Heilman
Date: November 18th, 2025
MAS.S62

Introduction



Bad news: We are all the goldfish

We going to look at the underlying assumptions of Bitcoin (and other cryptocurrencies) and investigate what happens when these assumptions are violated.

Good news:we can improve the design that keeps us from crashing to the floor

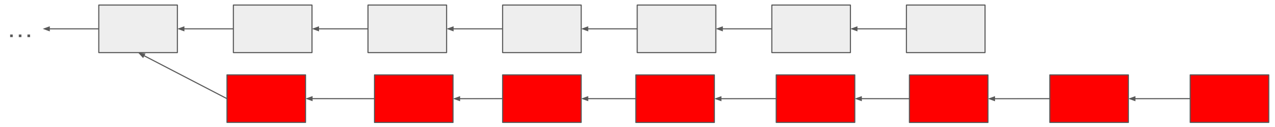
Dare we make our fish bowl bigger?

Things that can break

- Consensus/protocol assumptions
- Incentives
- Communication assumptions i.e. partitions
- Software
    - Bugs in wallets
    - Bugs in nodes
    - Bugs in protocols
- Cryptographic assumptions
    - Hash function breaks
    - Signature algorithms breaks
    - …

Consensus

PoW assumes 51% of the mining power behaves honestly*. What if it doesn't?



- An attacker controlling greater than >50 mining power can rewrite history, …allows double spending.
- Can't steal coins directly, because such an attacker can't forge signatures*.
- Can steal coins by reversing mining rewards and claim them as their own.

What happens if this assumption stops being violated? Is recovery possible?

Incentives (abridged.)

Incentives tend to be an assumption underlying other assumptions:

- Why should users not reveal their secret keys?
- Why should miners behave honestly?
- What if someone discovers that it is cheaper to mine empty blocks?

We can model incentives using game theory, economic rationality, legality,...

Compellence vs Deterrence?

…but hard to predict what people will do:
People don’t always act in their interests → Why do casinos exist?

Achilles, your choices are:
A. glory+death or B. a long life?
Glory, obv!

https://commons.wikimedia.org/wiki/File:Achilles_departure_Eretria_Painter_CdM_Paris_851.jpg

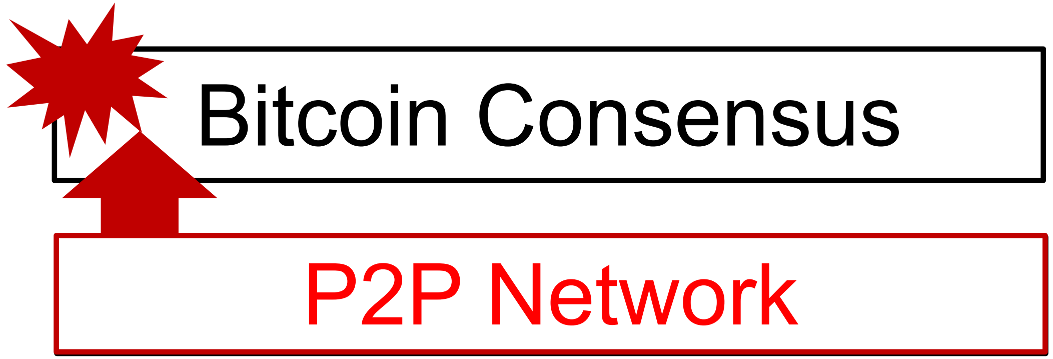
"I should choose, so I might live on earth as some penniless man, rather than to be lord over all the dead that have perished"
Achilles later (as a ghost)

https://www.britishmuseum.org/collection/object/G_1843-1103-61?selectedImageId=1613192781

Many important human stories are about regret over bad choices

Communication (Partitions)

Bitcoin is often thought to be secure as long as 51% of the mining power is honest, **…but** this assumes all parties see all valid blocks/transactions.

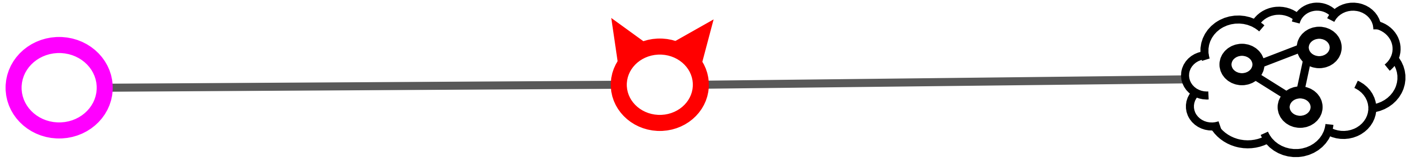


Bitcoin relies on its P2P network to deliver this information.

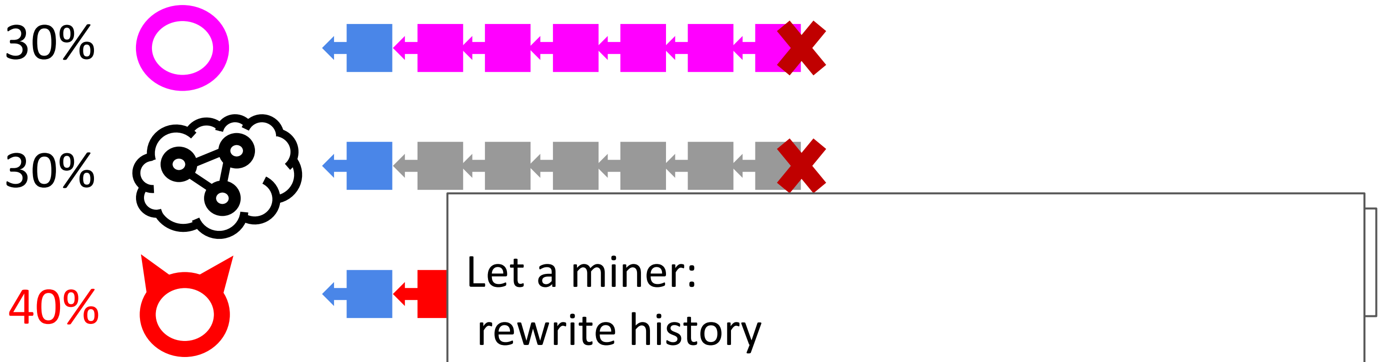
Control the P2P network □ control info flow □ control the blockchain.

Communication (Partitions)

Assume we have an attacker than can control what messages you can see



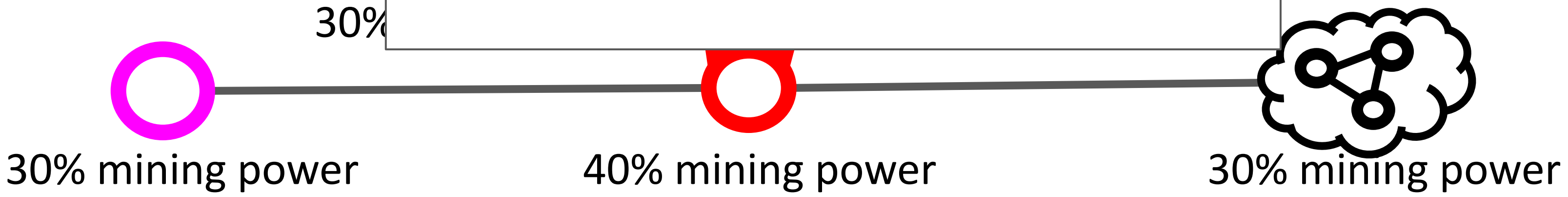
51% attacks with 40% mining power



Let a miner:

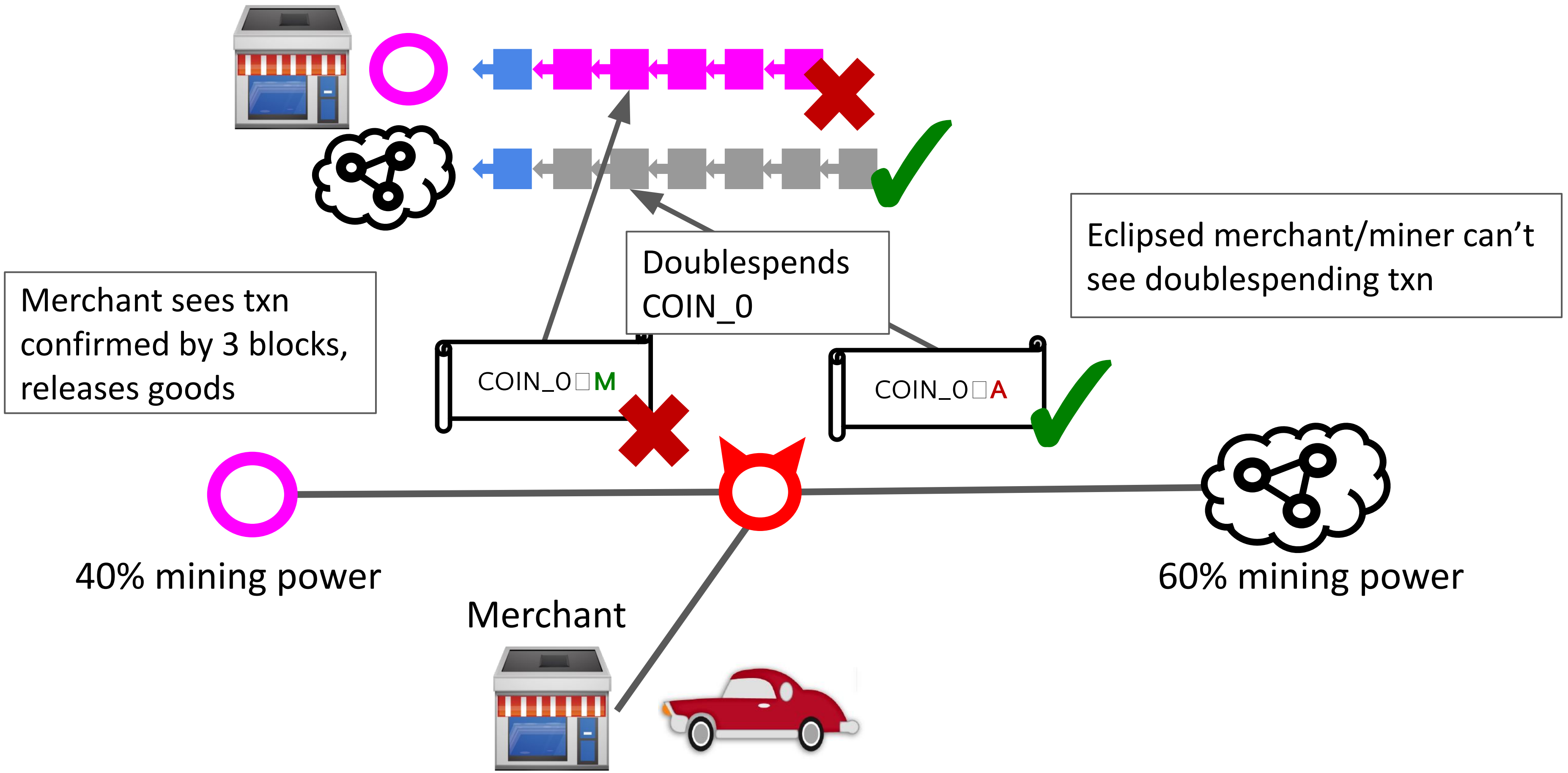
rewrite history

- Rewrite the history of the blockchain
- Censor transactions/blocks
- Take 100% of the block rewards/fees
- Better selfish mining attacks



Attacker then out competes each partitioned miner

N-Confirmation Double Spending


Merchant sees txn confirmed by 3 blocks, releases goods
Doublespends COIN_0
Eclipsed merchant/miner can't see doublespending txn
COIN_0M
COIN_0A
40% mining power
60% mining power
Merchant

Communication (Partitions)

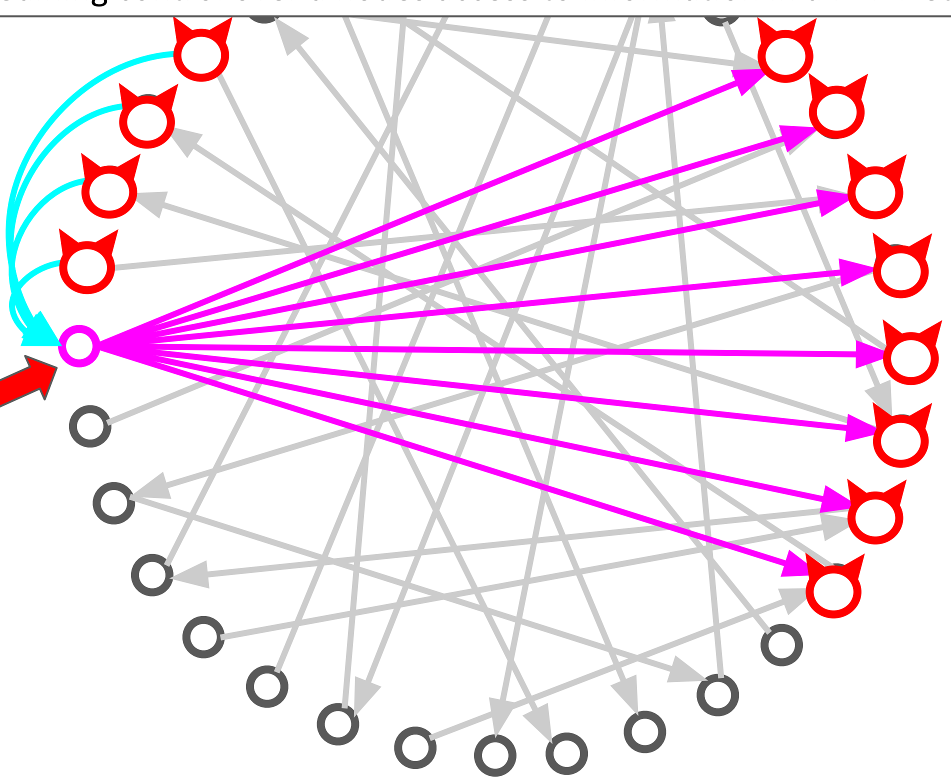
There are a few ways to accomplish this:

- Off-path (attacker manipulates the p2p network)

- In-path (attacker can sit in the middle between a node and the rest of the network)

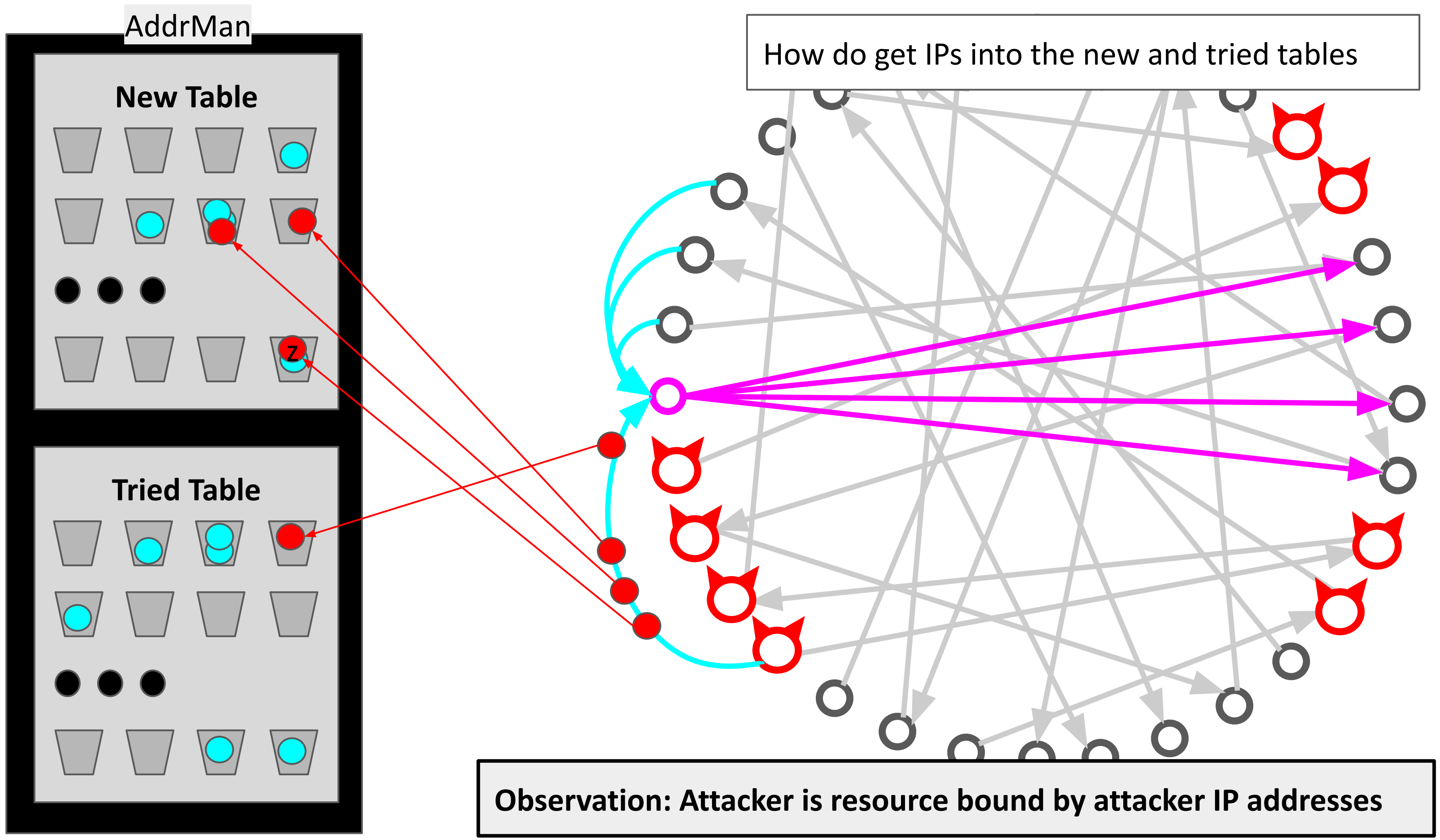
Communication (Partitions)

Information Eclipsing Attack (def):
Gaining control over a nodes access to information in a P2P Network



Surrounded by attacker nodes

2015, “Eclipse Attacks on Bitcoin’s Peer-to-Peer Network”

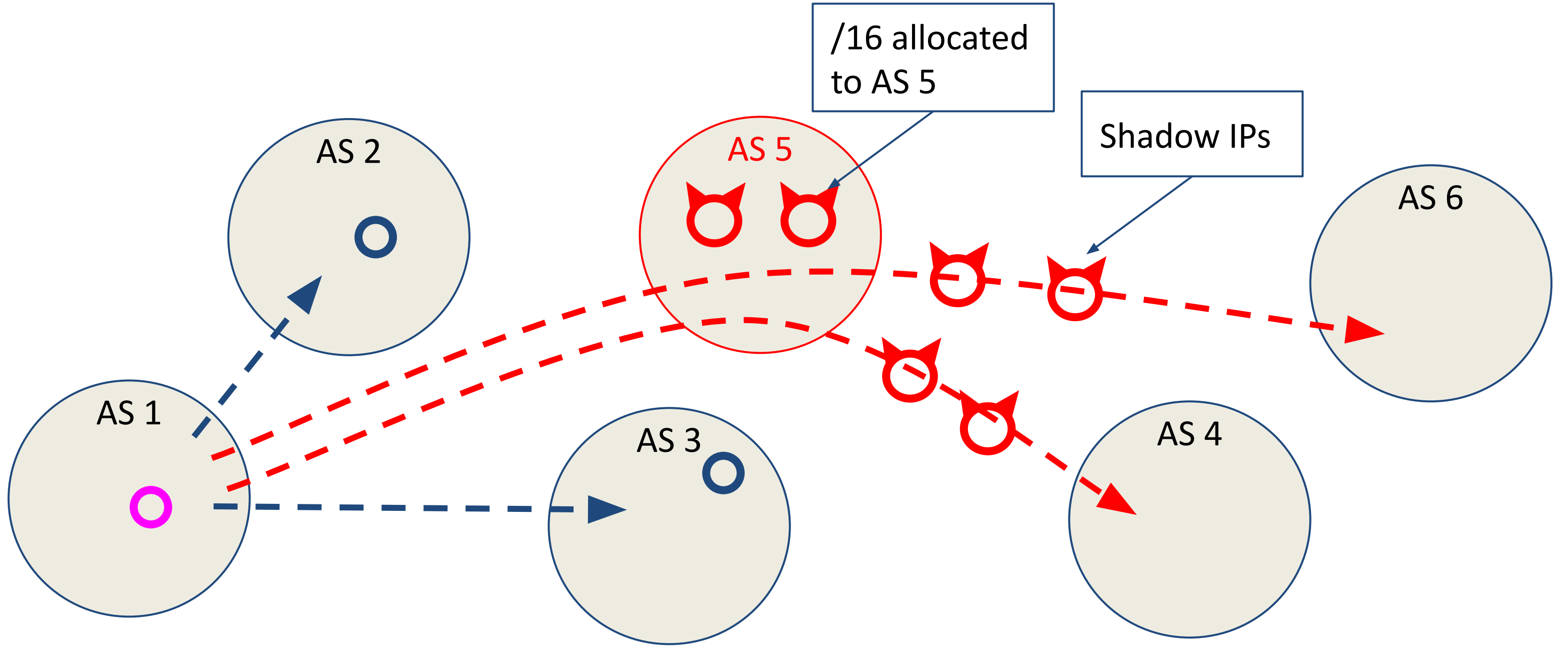


How do get IPs into the new and tried tables

Observation: Attacker is resource bound by attacker IP addresses

2020, Erebus “A Stealthier Partitioning Attack…”

“A Stealthier Partitioning Attack …” observes that an AS (Autonomous System) controls the IPs allocated to it **… and** controls IP that are routed through it



“A Stealthier Partitioning Attack against Bitcoin …”, M. Tran et. al. (2020)

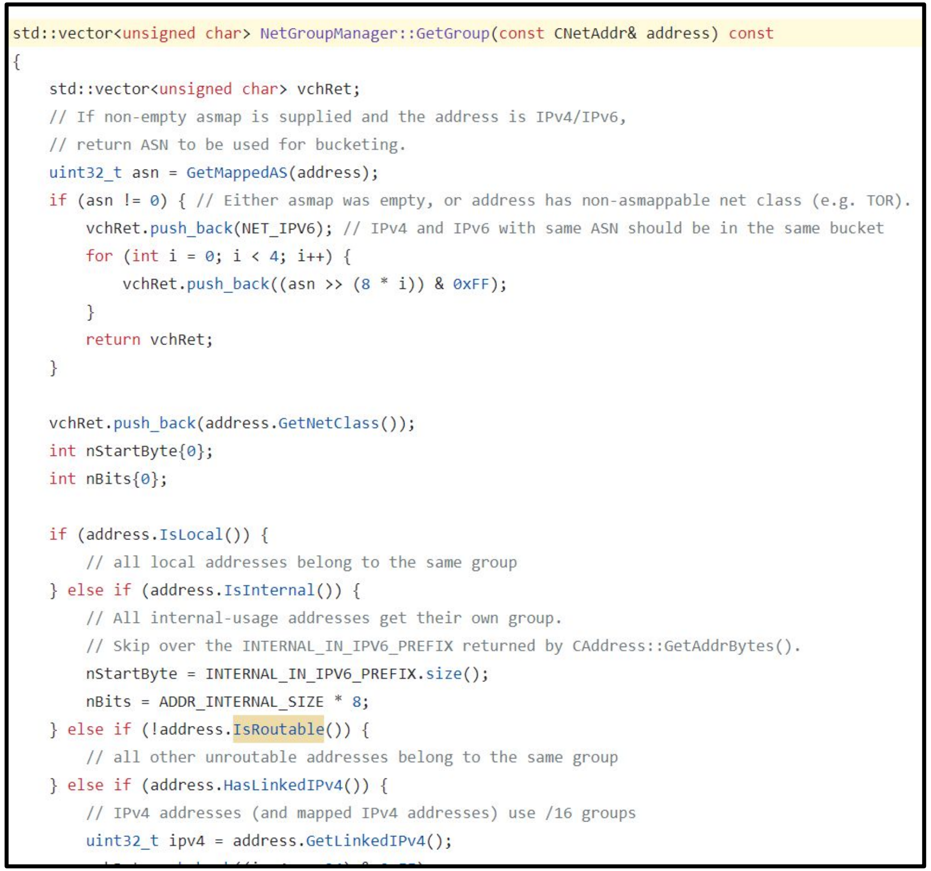
2020, Erebus “A Stealthier Partitioning Attack…”

Vulnerability - AS Topology != NetGroups:
NetGroups use IP allocation as proxy for diversity, but an AS can control more than their allocation

Countermeasure - AS-based NetGroups:
AS NetGroups that reflect AS Topology

NetGroups: Maps addresses to sets

```
IF ASMAP:
        IPv4/IPv6 = group by ASMAP
ELSE
        IPv4: netGroup = first 16 bits of IP address
        IPv6: netGroup = first 32 bits of IP address
        TOR: netGroup = first 4 bits of .Onion
        I2P: netGroup = first 4 bits of I2P address
```



```
std::vector<unsigned char> NetGroupManager::GetGroup(const CNetAddr& address) const
{
    std::vector<unsigned char> vchRet;
    // If non-empty asmap is supplied and the address is IPv4/IPv6,
    // return ASN to be used for bucketing.
    uint32_t asn = GetMappedAS(address);
    if (asn != 0) { // Either asmap was empty, or address has non-asmappable net class (e.g. TOR).
        vchRet.push_back(NET_IPV6); // IPv4 and IPv6 with same ASN should be in the same bucket
        for (int i = 0; i < 4; i++) {
            vchRet.push_back((asn >> (8 * i)) & 0xFF);
        }
        return vchRet;
    }

    vchRet.push_back(address.GetNetClass());
    int nStartByte{0};
    int nBits{0};

    if (address.IsLocal()) {
        // all local addresses belong to the same group
    } else if (address.IsInternal()) {
        // All internal-usage addresses get their own group.
        // Skip over the INTERNAL_IN_IPV6_PREFIX returned by CAddress::GetAddrBytes().
        nStartByte = INTERNAL_IN_IPV6_PREFIX.size();
        nBits = ADDR_INTERNAL_SIZE * 8;
    } else if (!address.IsRoutable()) {
        // all other unroutable addresses belong to the same group
    } else if (address.HasLinkedIPv4()) {
        // IPv4 addresses (and mapped IPv4 addresses) use /16 groups
        uint32_t ipv4 = address.GetLinkedIPv4();
```

https://github.com/bitcoin/bitcoin/blob/c1106cfef514115e91c2185a58840d7fb0e34c89/src/netgroup.cpp#L17

Countermeasures and Improvements over time

	Proposed	Merge	PR	Ver	Benefit
Deterministic Random Eviction	2015	2015	PR #5941	0.10	Fixes: Try-try-again, Eviction Bias
Random Selection	2015	2015	PR #5941	0.10	Fixes: Selection Bias
Feeler Connections	2015	2016	PR #8282	0.14	More online IPs in tried
Only Add Out Conns to Tried		2016	PR #8594	0.14	Fewer attacker IPs in tried (per time)
Test-before-Evict	2015	2018	PR #9037	0.13	Attacker can't evict honest IPs
Discourage rather than ban		2019	PR #14929	0.18	Fixes: TOR exit node ban
AS-based NetGroups	2019	2020	PR #16702	0.20	Fixes: AS Topology != NetGroups
Anchor Connections	2015	2020	PR #17428	0.21	Impedes Restart-based Eclipse attacks

I'm covering only a small amount of the attacks and countermeasures

Communication (Partitions)

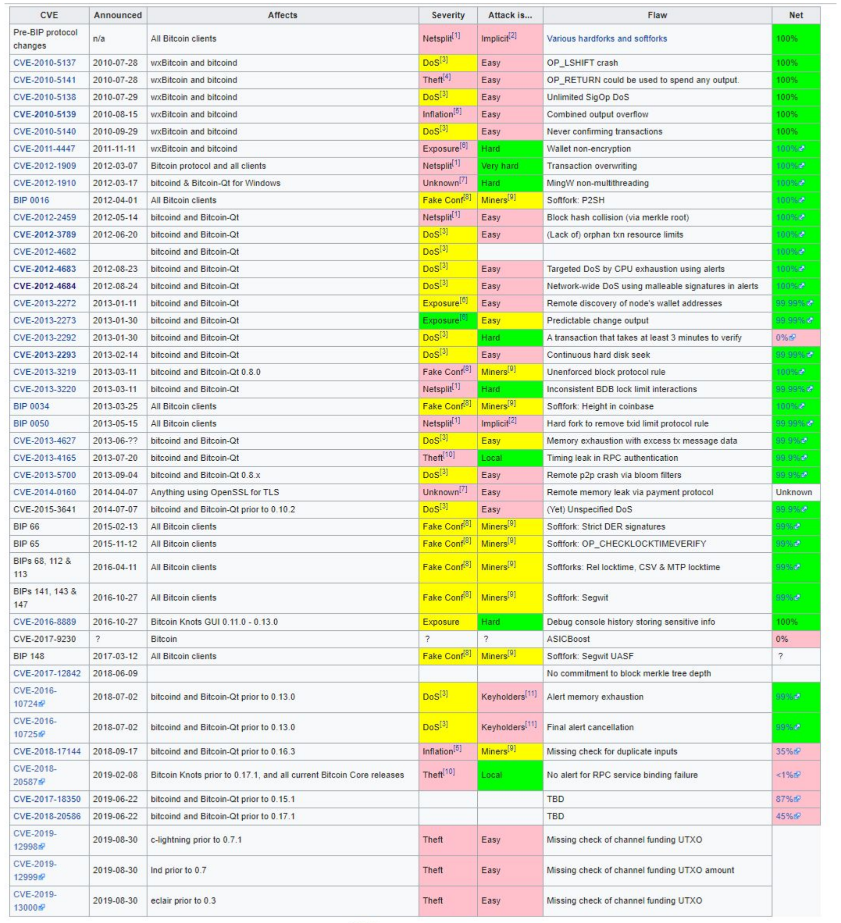
I'm covering only a small amount of partition attacks and countermeasures.

Software bugs

All protocols rely on the software that runs them being correct, but software is buggy.

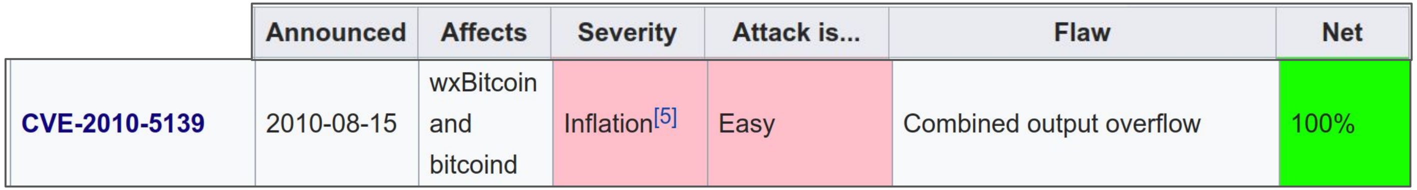
Every other assumption can be violated due to software bugs:

- Consensus splits
- Unfair mining
- Signing keys leaking

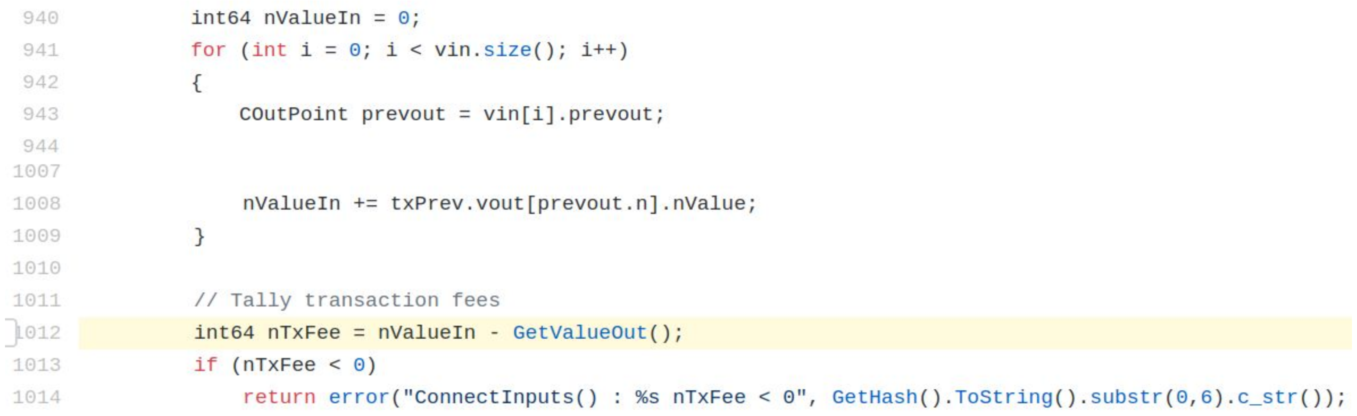


CVE	Announced	Affects	Severity	Attack is...	Flaw	Net
Pre-BIP protocol changes	n/a	All Bitcoin clients	Netsplit[1]	Implicit[2]	Various hardforks and softforks	100%
CVE-2010-5137	2010-07-28	wxBitcoin and bitcoind	DoS[3]	Easy	OP_LSHIFT crash	100%
CVE-2010-5141	2010-07-28	wxBitcoin and bitcoind	Theft[4]	Easy	OP_RETURN could be used to spend any output.	100%
CVE-2010-5138	2010-07-29	wxBitcoin and bitcoind	DoS[3]	Easy	Unlimited SigOp DoS	100%
CVE-2010-5139	2010-08-15	wxBitcoin and bitcoind	Inflation[5]	Easy	Combined output overflow	100%
CVE-2010-5140	2010-09-29	wxBitcoin and bitcoind	DoS[3]	Easy	Never confirming transactions	100%
CVE-2011-4447	2011-11-11	wxBitcoin and bitcoind	Exposure[6]	Hard	Wallet non-encryption	100%
CVE-2012-1909	2012-03-07	Bitcoin protocol and all clients	Netsplit[1]	Very hard	Transaction overwriting	100%
CVE-2012-1910	2012-03-17	bitcoind & Bitcoin-Qt for Windows	Unknown[7]	Hard	MingW non-multithreading	100%
BIP 0016	2012-04-01	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: P2SH	100%
CVE-2012-2459	2012-05-14	bitcoind and Bitcoin-Qt	Netsplit[1]	Easy	Block hash collision (via merkle root)	100%
CVE-2012-3789	2012-06-20	bitcoind and Bitcoin-Qt	DoS[3]	Easy	(Lack of) orphan txn resource limits	100%
CVE-2012-4682		bitcoind and Bitcoin-Qt	DoS[3]			100%
CVE-2012-4683	2012-08-23	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Targeted DoS by CPU exhaustion using alerts	100%
CVE-2012-4684	2012-08-24	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Network-wide DoS using malleable signatures in alerts	100%
CVE-2013-2272	2013-01-11	bitcoind and Bitcoin-Qt	Exposure[6]	Easy	Remote discovery of node's wallet addresses	99.99%
CVE-2013-2273	2013-01-30	bitcoind and Bitcoin-Qt	Exposure[6]	Easy	Predictable change output	99.99%
CVE-2013-2292	2013-01-30	bitcoind and Bitcoin-Qt	DoS[3]	Hard	A transaction that takes at least 3 minutes to verify	0%
CVE-2013-2293	2013-02-14	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Continuous hard disk seek	99.99%
CVE-2013-3219	2013-03-11	bitcoind and Bitcoin-Qt 0.8.0	Fake Conf[8]	Miners[9]	Unenforced block protocol rule	100%
CVE-2013-3220	2013-03-11	bitcoind and Bitcoin-Qt	Netsplit[1]	Hard	Inconsistent BDB lock limit interactions	99.99%
BIP 0034	2013-03-25	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Height in coinbase	100%
BIP 0050	2013-05-15	All Bitcoin clients	Netsplit[1]	Implicit[2]	Hard fork to remove txid limit protocol rule	99.99%
CVE-2013-4627	2013-06-??	bitcoind and Bitcoin-Qt	DoS[3]	Easy	Memory exhaustion with excess tx message data	99.9%
CVE-2013-4165	2013-07-20	bitcoind and Bitcoin-Qt	Theft[10]	Local	Timing leak in RPC authentication	99.9%
CVE-2013-5700	2013-09-04	bitcoind and Bitcoin-Qt 0.8.x	DoS[3]	Easy	Remote p2p crash via bloom filters	99.9%
CVE-2014-0160	2014-04-07	Anything using OpenSSL for TLS	Unknown[7]	Easy	Remote memory leak via payment protocol	Unknown
CVE-2015-3641	2014-07-07	bitcoind and Bitcoin-Qt prior to 0.10.2	DoS[3]	Easy	(Yet) Unspecified DoS	99.9%
BIP 66	2015-02-13	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Strict DER signatures	99%
BIP 65	2015-11-12	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: OP_CHECKLOCKTIMEVERIFY	99%
BIPs 68, 112 & 113	2016-04-11	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softforks: Rel locktime, CSV & MTP locktime	99%
BIPs 141, 143 & 147	2016-10-27	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Segwit	99%
CVE-2016-8889	2016-10-27	Bitcoin Knots GUI 0.11.0 - 0.13.0	Exposure	Hard	Debug console history storing sensitive info	100%
CVE-2017-9230	?	Bitcoin	?	?	ASICBoost	0%
BIP 148	2017-03-12	All Bitcoin clients	Fake Conf[8]	Miners[9]	Softfork: Segwit UASF	?
CVE-2017-12842	2018-06-09				No commitment to block merkle tree depth	
CVE-2016-10724	2018-07-02	bitcoind and Bitcoin-Qt prior to 0.13.0	DoS[3]	Keyholders[11]	Alert memory exhaustion	99%
CVE-2016-10725	2018-07-02	bitcoind and Bitcoin-Qt prior to 0.13.0	DoS[3]	Keyholders[11]	Final alert cancellation	99%
CVE-2018-17144	2018-09-17	bitcoind and Bitcoin-Qt prior to 0.16.3	Inflation[5]	Miners[9]	Missing check for duplicate inputs	35%
CVE-2018-20587	2019-02-08	Bitcoin Knots prior to 0.17.1, and all current Bitcoin Core releases	Theft[10]	Local	No alert for RPC service binding failure	<1%
CVE-2017-18350	2019-06-22	bitcoind and Bitcoin-Qt prior to 0.15.1			TBD	87%
CVE-2018-20586	2019-06-22	bitcoind and Bitcoin-Qt prior to 0.17.1			TBD	45%
CVE-2019-12998	2019-08-30	c-lightning prior to 0.7.1	Theft	Easy	Missing check of channel funding UTXO	
CVE-2019-12999	2019-08-30	lnd prior to 0.7	Theft	Easy	Missing check of channel funding UTXO amount	
CVE-2019-13000	2019-08-30	eclair prior to 0.3	Theft	Easy	Missing check of channel funding UTXO	

CVE-2010-5139: Inflation Bug



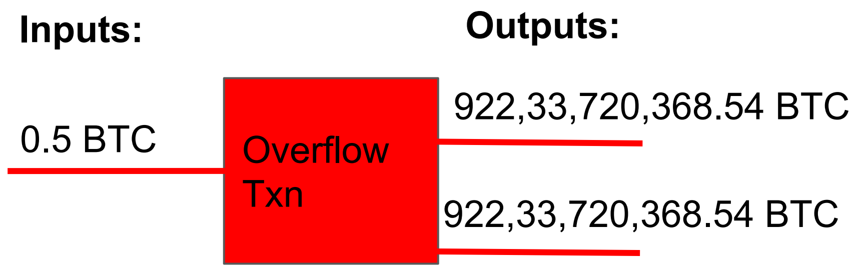
	Announced	Affects	Severity	Attack is...	Flaw	Net
CVE-2010-5139	2010-08-15	wxBitcoin and bitcoind	Inflation[5]	Easy	Combined output overflow	100%



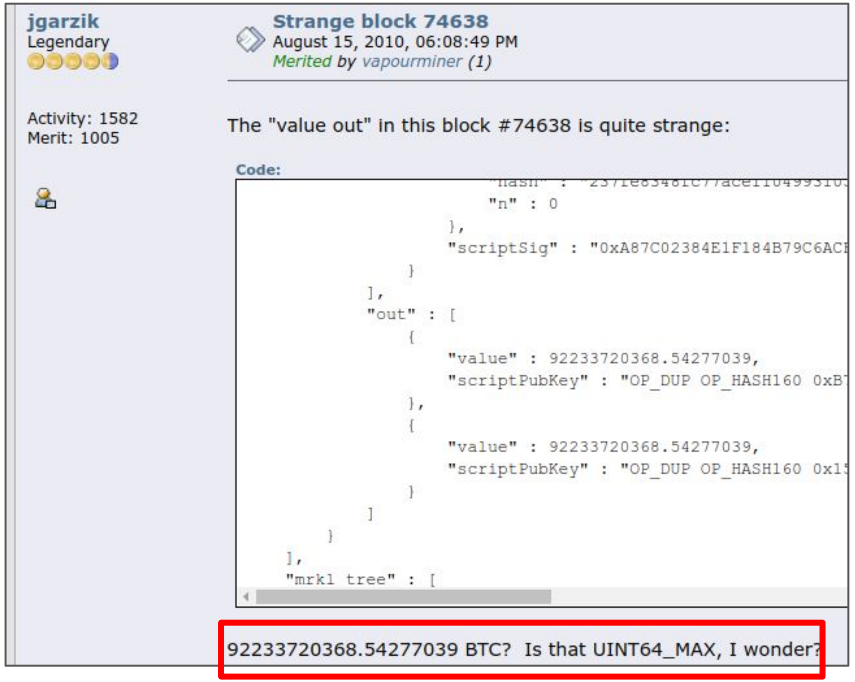
```
940          int64 nValueIn = 0;
941          for (int i = 0; i < vin.size(); i++)
942          {
943              COutPoint prevout = vin[i].prevout;
944
1007
1008             nValueIn += txPrev.vout[prevout.n].nValue;
1009         }
1010
1011         // Tally transaction fees
1012         int64 nTxFee = nValueIn - GetValueOut();
1013         if (nTxFee < 0)
1014             return error("ConnectInputs() : %s nTxFee < 0", GetHash().ToString().substr(0,6).c_str());
```

https://github.com/bitcoin/bitcoin/blob/4bd188c4383d6e614e18f79dc337fbabe8464c82/main.cpp#L1012

CVE-2010-5139: Inflation Bug



```
  922,33,720,368.54
+ 922,33,720,368.54
---------------------------
= -0.1 BTC
```



jgarzik
Legendary

Activity: 1582
Merit: 1005

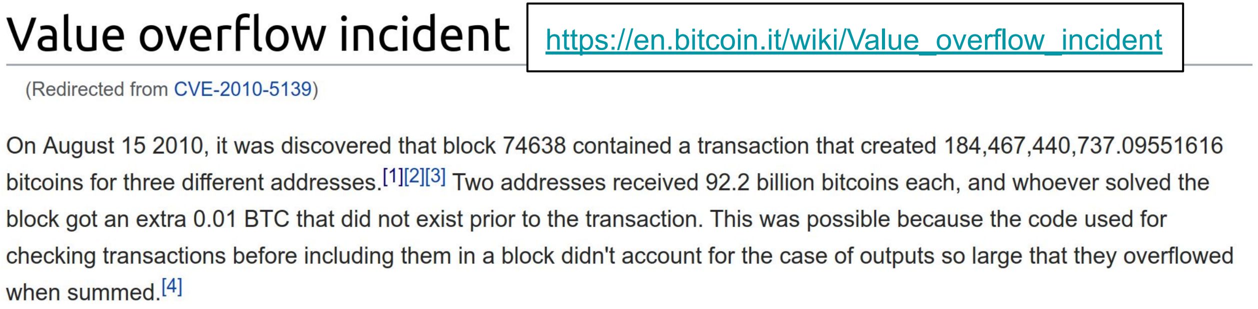
Strange block 74638
August 15, 2010, 06:08:49 PM
Merited by vapourminer (1)

The "value out" in this block #74638 is quite strange:

Code:
```
                          "hash" : "237fe83481c77ace110499310
                          "n" : 0
                      },
                      "scriptSig" : "0xA87C02384E1F184B79C6ACE
                  }
              ],
              "out" : [
                  {
                      "value" : 92233720368.54277039,
                      "scriptPubKey" : "OP_DUP OP_HASH160 0xB7
                  },
                  {
                      "value" : 92233720368.54277039,
                      "scriptPubKey" : "OP_DUP OP_HASH160 0x1
                  }
              ]
          }
      ],
      "mrkl_tree" : [
```

92233720368.54277039 BTC? Is that UINT64_MAX, I wonder?

CVE-2010-5139: Inflation Bug

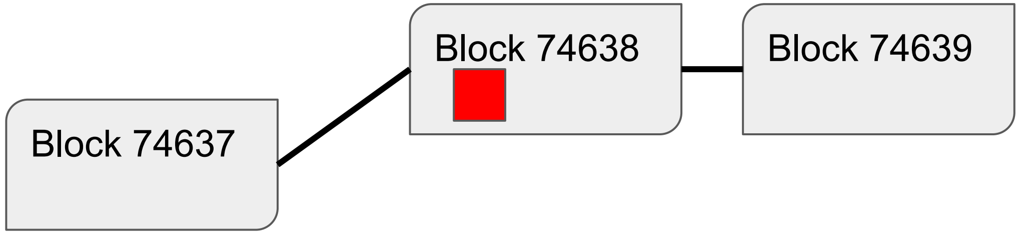


Value overflow incident

https://en.bitcoin.it/wiki/Value_overflow_incident

(Redirected from CVE-2010-5139)

On August 15 2010, it was discovered that block 74638 contained a transaction that created 184,467,440,737.09551616 bitcoins for three different addresses.[1][2][3] Two addresses received 92.2 billion bitcoins each, and whoever solved the block got an extra 0.01 BTC that did not exist prior to the transaction. This was possible because the code used for checking transactions before including them in a block didn't account for the case of outputs so large that they overflowed when summed.[4]



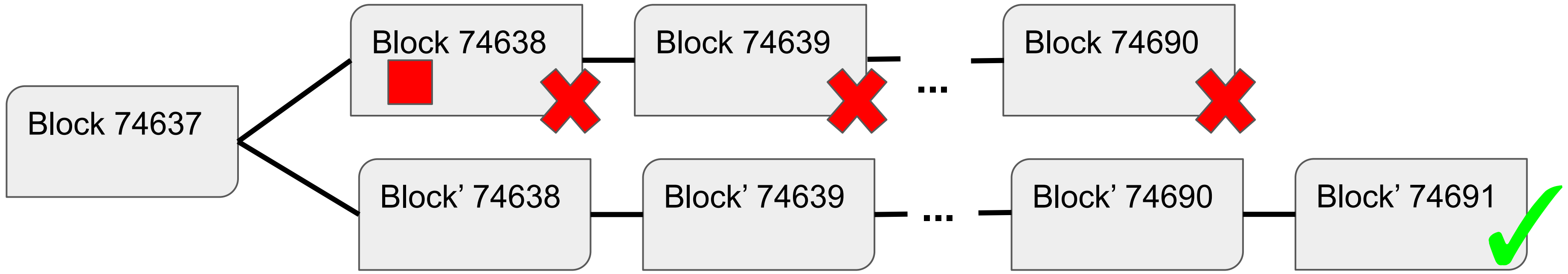
At this point there were several blocks which recorded that two addresses contained 92.2 Billion Bitcoins

CVE-2010-5139: Inflation Bug

Overflow was fixed via patch that softforked the bug.

This resulted in two chains:

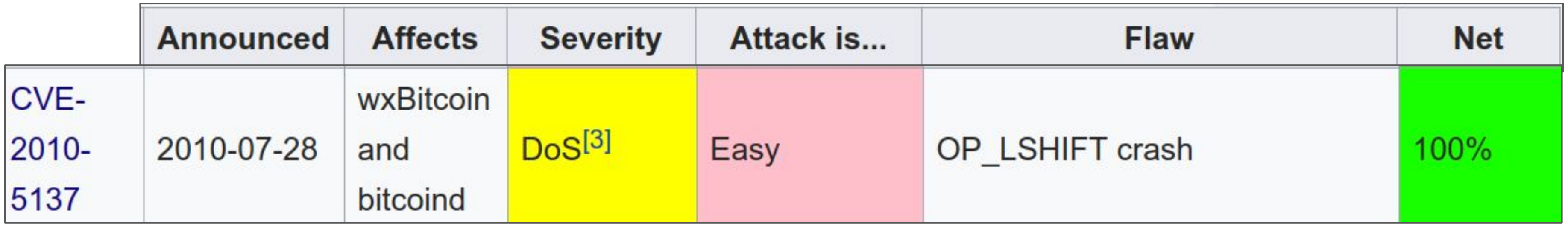
1. The buggy chain with the overflow transaction
2. The chain with the patch



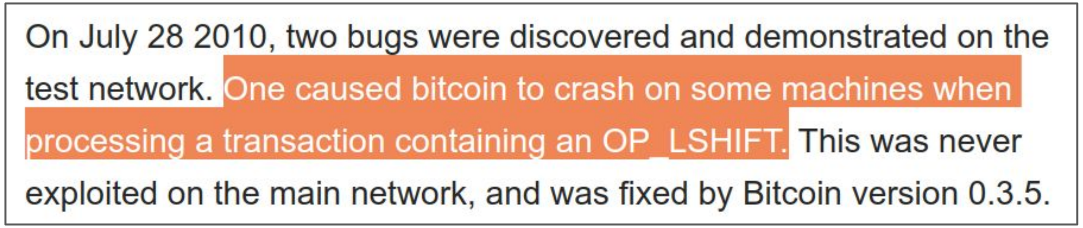
The patched chain had more mining power and became the longest chain,

...unpatched nodes then switched to the longest chain

CVE-2010-5137: Crash Bug



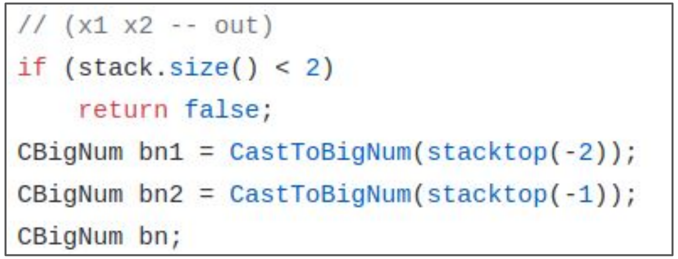
	Announced	Affects	Severity	Attack is...	Flaw	Net
CVE-2010-5137	2010-07-28	wxBitcoin and bitcoind	DoS[3]	Easy	OP_LSHIFT crash	100%



On July 28 2010, two bugs were discovered and demonstrated on the test network. One caused bitcoin to crash on some machines when processing a transaction containing an OP_LSHIFT. This was never exploited on the main network, and was fixed by Bitcoin version 0.3.5.

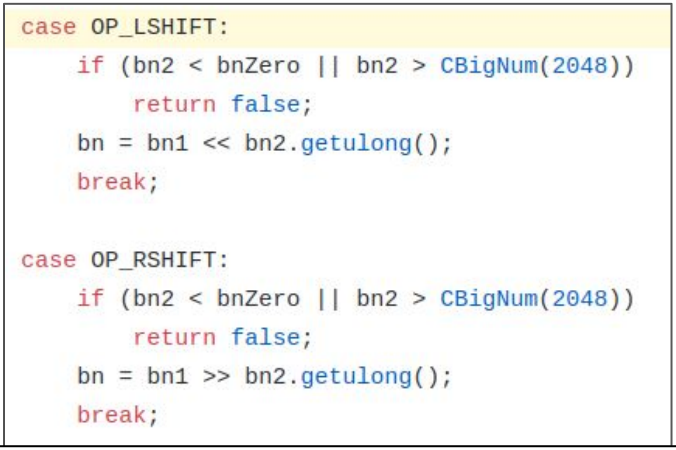
Crash Bug (in BTC)

On July 28 2010, two bugs were discovered and demonstrated on the test network. One caused bitcoin to crash on some machines when processing a transaction containing an OP_LSHIFT. This was never exploited on the main network, and was fixed by Bitcoin version 0.3.5.



```
// (x1 x2 -- out)
if (stack.size() < 2)
    return false;
CBigNum bn1 = CastToBigNum(stacktop(-2));
CBigNum bn2 = CastToBigNum(stacktop(-1));
CBigNum bn;
```

...

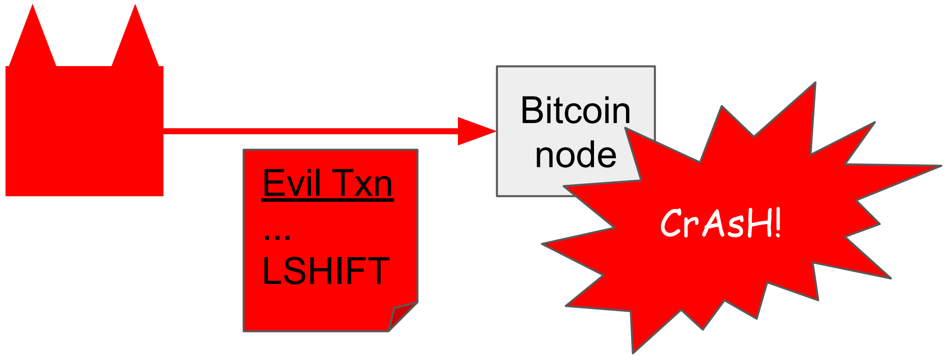


```
case OP_LSHIFT:
    if (bn2 < bnZero || bn2 > CBigNum(2048))
        return false;
    bn = bn1 << bn2.getulong();
    break;

case OP_RSHIFT:
    if (bn2 < bnZero || bn2 > CBigNum(2048))
        return false;
    bn = bn1 >> bn2.getulong();
    break;
```

Can you spot the vulnerability?

https://github.com/bitcoin/bitcoin/blob/0a61b0df1224a5470bcddab302bc199ca5a9e356/script.cpp#L636

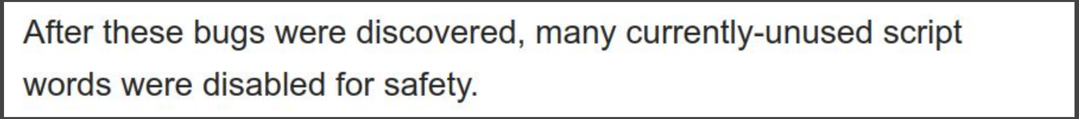
Crash Bug (in BTC)



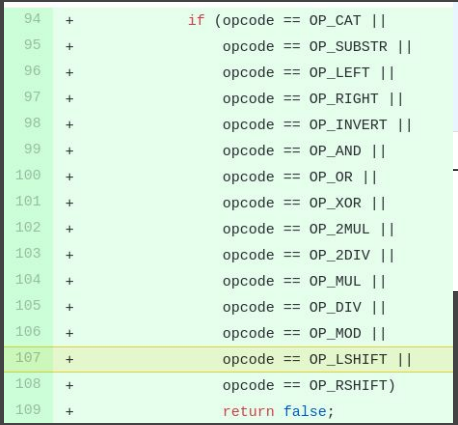
Impact:

- CVEs say it only crashed on some machines
  - worse than crashing all nodes?
- What happens if 40% of network fixes this bug **...and** evil txn is included in a block?

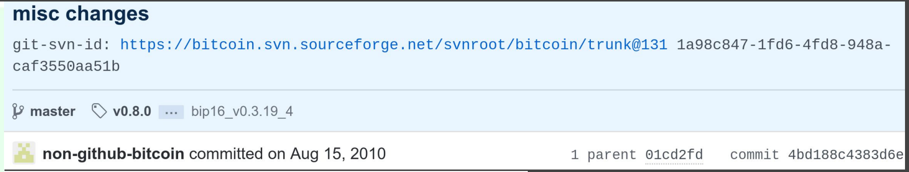
CVE-2010-5137: OP_LSHIFT Crash (Fix)



After these bugs were discovered, many currently-unused script words were disabled for safety.



```
+            if (opcode == OP_CAT ||
+                opcode == OP_SUBSTR ||
+                opcode == OP_LEFT ||
+                opcode == OP_RIGHT ||
+                opcode == OP_INVERT ||
+                opcode == OP_AND ||
+                opcode == OP_OR ||
+                opcode == OP_XOR ||
+                opcode == OP_2MUL ||
+                opcode == OP_2DIV ||
+                opcode == OP_MUL ||
+                opcode == OP_DIV ||
+                opcode == OP_MOD ||
+                opcode == OP_LSHIFT ||
+                opcode == OP_RSHIFT)
+                return false;
```



misc changes

git-svn-id: https://bitcoin.svn.sourceforge.net/svnroot/bitcoin/trunk@131 1a98c847-1fd6-4fd8-948a-caf3550aa51b

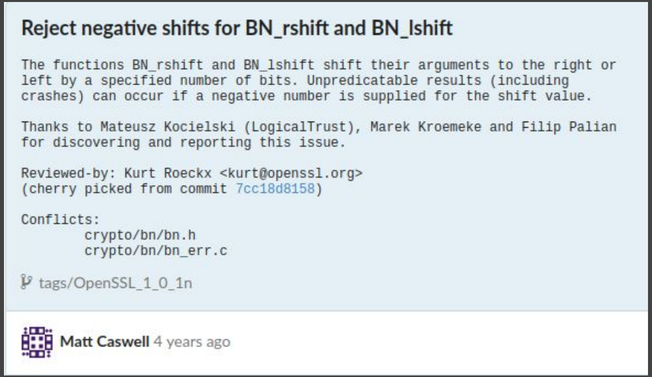
master v0.8.0 … bip16_v0.3.19_4

non-github-bitcoin committed on Aug 15, 2010 1 parent 01cd2fd commit 4bd188c4383d6e

https://github.com/bitcoin/bitcoin/commit/4bd188c4383d6e614e18f79dc337fbabe8464c82#diff-8458adcedc17d046942185cb709ff5c3R107

What would have happened if they had just fixed the bug … and not disabled the OP_LSHIFT?

CVE-2010-5137: OP_LSHIFT Crash (A theory)

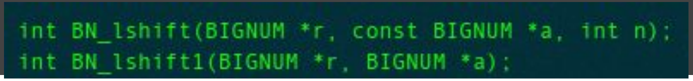


Reject negative shifts for BN_rshift and BN_lshift

```
The functions BN_rshift and BN_lshift shift their arguments to the right or
left by a specified number of bits. Unpredicatable results (including
crashes) can occur if a negative number is supplied for the shift value.

Thanks to Mateusz Kocielski (LogicalTrust), Marek Kroemeke and Filip Palian
for discovering and reporting this issue.

Reviewed-by: Kurt Roeckx <kurt@openssl.org>
(cherry picked from commit 7cc18d8158)

Conflicts:
        crypto/bn/bn.h
        crypto/bn/bn_err.c
```

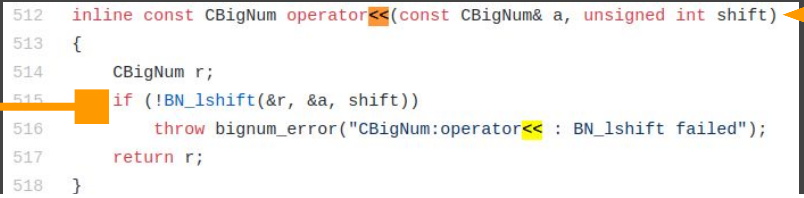
tags/OpenSSL_1_0_1n

Matt Caswell 4 years ago



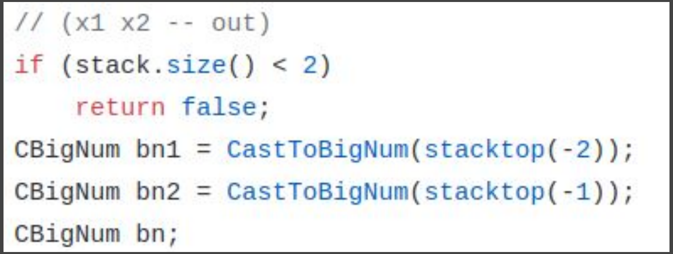
```
int BN_lshift(BIGNUM *r, const BIGNUM *a, int n);
int BN_lshift1(BIGNUM *r, BIGNUM *a);
```

https://docs.huihoo.com/doxygen/openssl/1.0.1c/bn__shift_8c.html



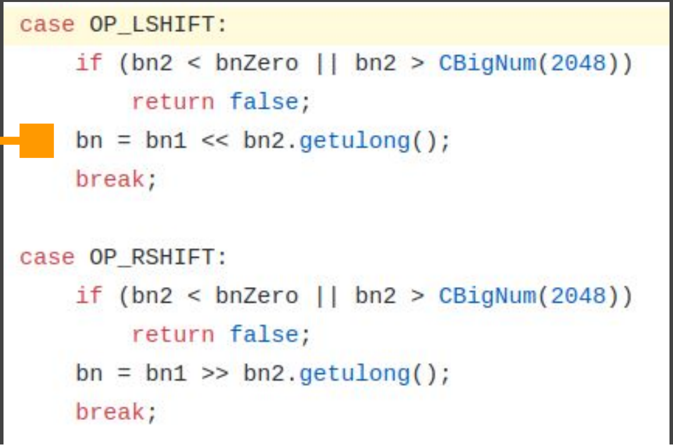
```
inline const CBigNum operator<<(const CBigNum& a, unsigned int shift)
{
    CBigNum r;
    if (!BN_lshift(&r, &a, shift))
        throw bignum_error("CBigNum:operator<< : BN_lshift failed");
    return r;
}
```

https://github.com/bitcoin/bitcoin/blob/0a61b0df1224a5470bcddab302bc199ca5a9e356/bignum.h#L512



```
// (x1 x2 -- out)
if (stack.size() < 2)
    return false;
CBigNum bn1 = CastToBigNum(stacktop(-2));
CBigNum bn2 = CastToBigNum(stacktop(-1));
CBigNum bn;
```

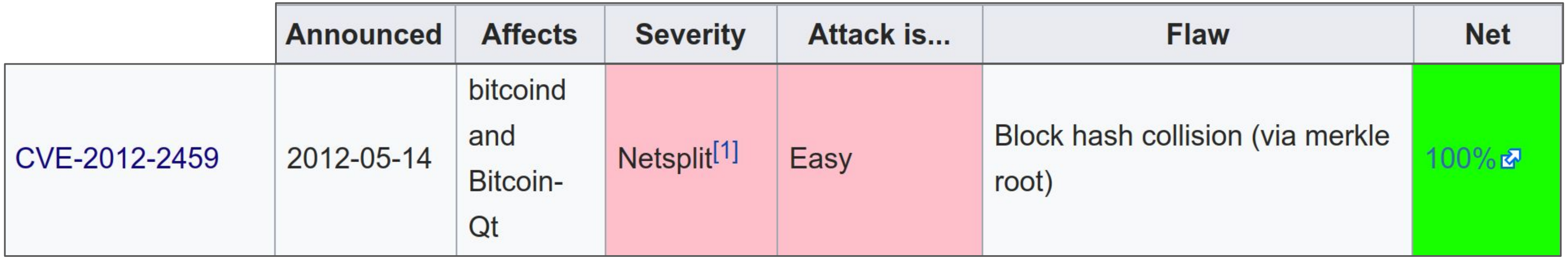
...



```
case OP_LSHIFT:
    if (bn2 < bnZero || bn2 > CBigNum(2048))
        return false;
    bn = bn1 << bn2.getulong();
    break;

case OP_RSHIFT:
    if (bn2 < bnZero || bn2 > CBigNum(2048))
        return false;
    bn = bn1 >> bn2.getulong();
    break;
```

https://github.com/bitcoin/bitcoin/blob/0a61b0df1224a5470bcddab302bc199ca5a9e356/script.cpp#L636

Netsplit (in BTC)



	Announced	Affects	Severity	Attack is...	Flaw	Net
CVE-2012-2459	2012-05-14	bitcoind and Bitcoin-Qt	Netsplit[1]	Easy	Block hash collision (via merkle root)	100%

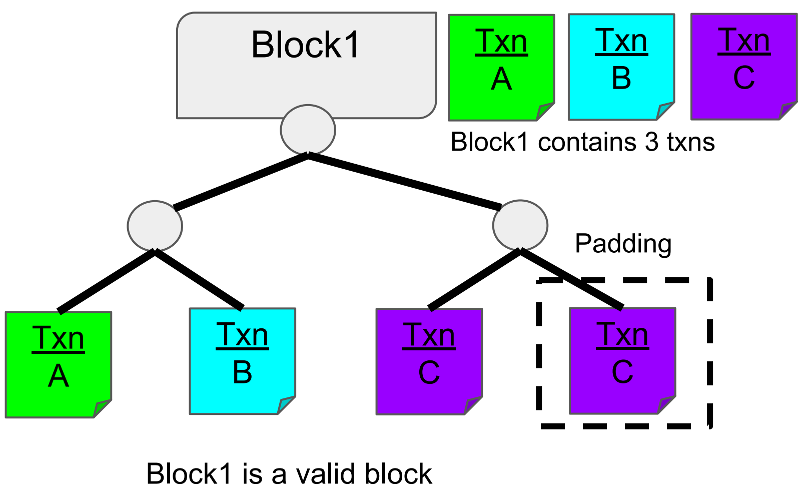
Exploits the fact that two semantically different blocks result in the same hash because the representation of the blocks collide

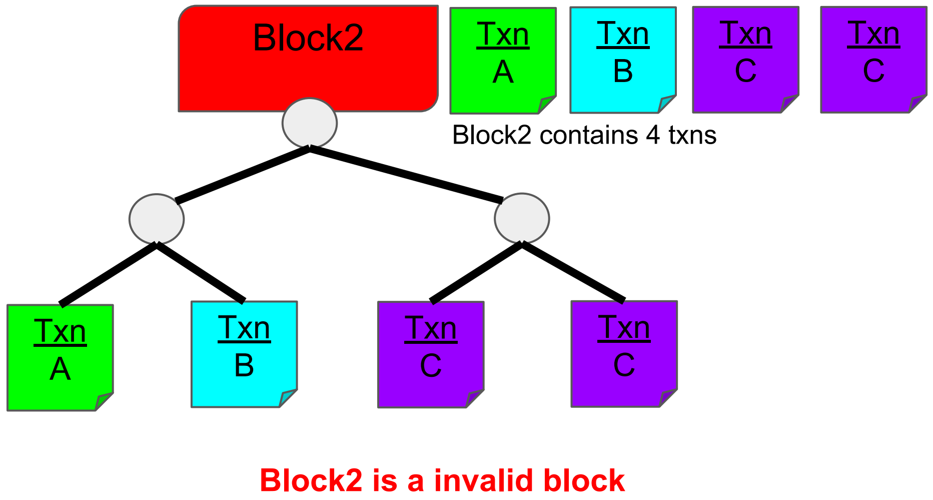
Block1 != **Block2**

Block1.repr = **Block2**.repr

H(**Block1**.repr) = H(**Block2**.repr)

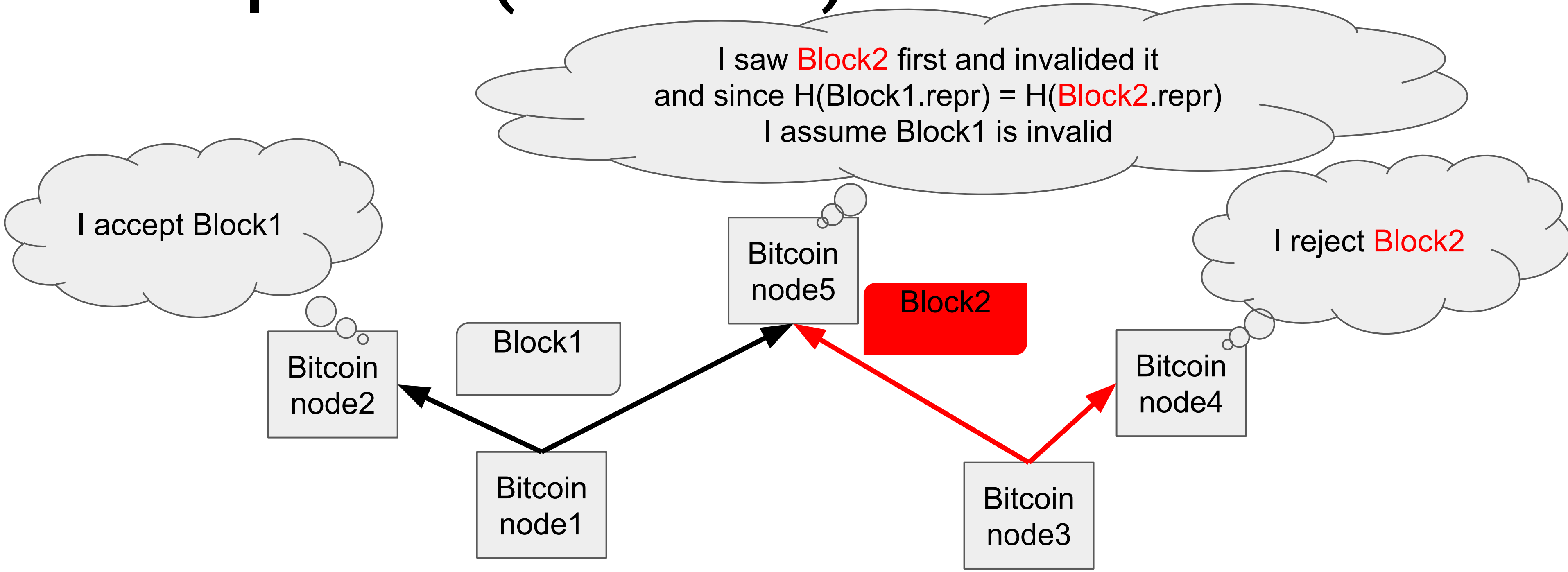
Netsplit (in BTC)





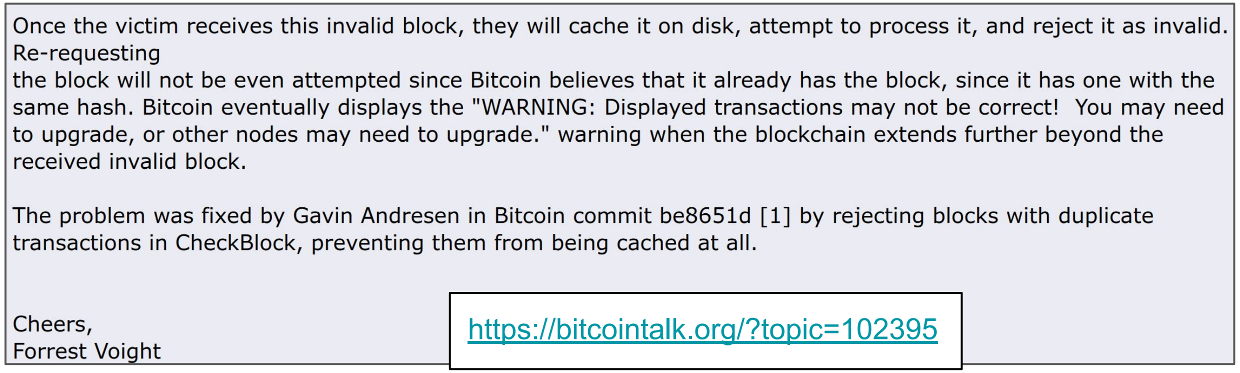
Block1 padding results in the same bytes as Block2 has TxnC twice.
Thus, Block1.repr = Block2.repr

Netsplit (in BTC)



Nodes that see Block1 first will accept Block1
Nodes that see Block2 first will reject Block2 and Block1
This will cause the network to split into nodes that either accept/reject Block1

Netsplit (in BTC)

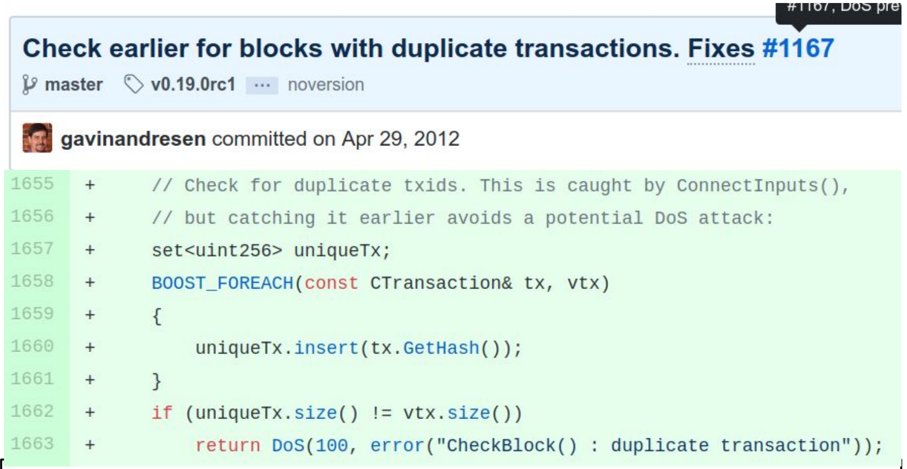


Once the victim receives this invalid block, they will cache it on disk, attempt to process it, and reject it as invalid. Re-requesting
the block will not be even attempted since Bitcoin believes that it already has the block, since it has one with the same hash. Bitcoin eventually displays the "WARNING: Displayed transactions may not be correct! You may need to upgrade, or other nodes may need to upgrade." warning when the blockchain extends further beyond the received invalid block.

The problem was fixed by Gavin Andresen in Bitcoin commit be8651d [1] by rejecting blocks with duplicate transactions in CheckBlock, preventing them from being cached at all.

Cheers,
Forrest Voight

https://bitcointalk.org/?topic=102395



Check earlier for blocks with duplicate transactions. Fixes #1167

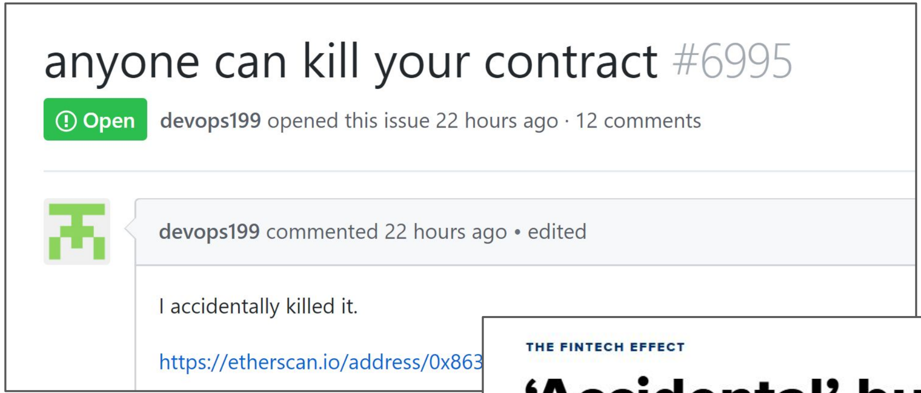
master · v0.19.0rc1 ··· noversion

gavinandresen committed on Apr 29, 2012

```
+     // Check for duplicate txids. This is caught by ConnectInputs(),
+     // but catching it earlier avoids a potential DoS attack:
+     set<uint256> uniqueTx;
+     BOOST_FOREACH(const CTransaction& tx, vtx)
+     {
+         uniqueTx.insert(tx.GetHash());
+     }
+     if (uniqueTx.size() != vtx.size())
+         return DoS(100, error("CheckBlock() : duplicate transaction"));
```

https://github.com/bitcoin/bitcoin/commit/be8651dde7b59e50e8c443da71c706667803d06d

Multisig Wallet bug (in ETH)



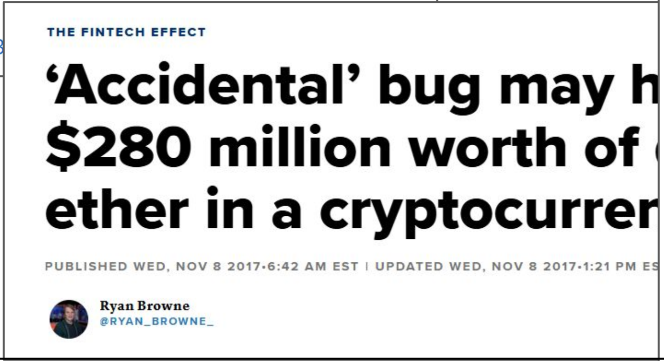
anyone can kill your contract #6995

Open devops199 opened this issue 22 hours ago · 12 comments

devops199 commented 22 hours ago • edited

I accidentally killed it.

https://etherscan.io/address/0x863

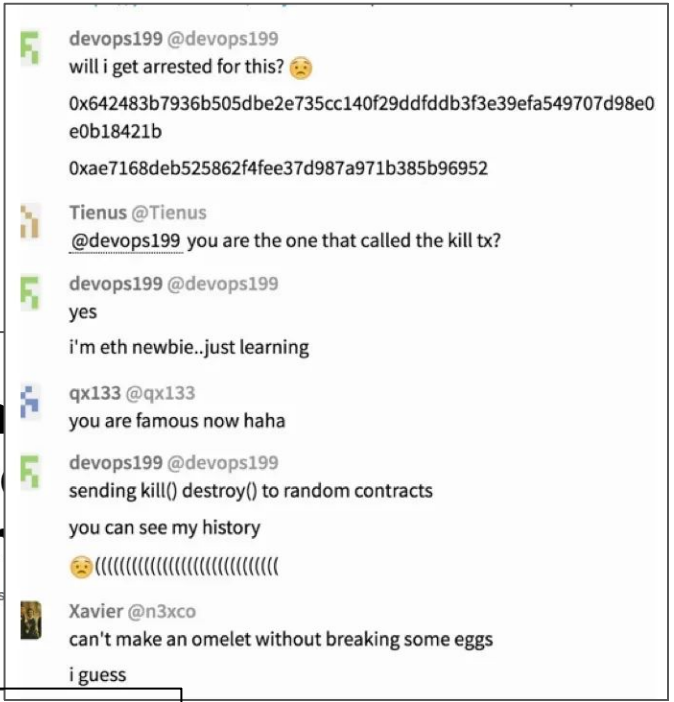


THE FINTECH EFFECT

'Accidental' bug may h
$280 million worth of
ether in a cryptocurren

PUBLISHED WED, NOV 8 2017·6:42 AM EST | UPDATED WED, NOV 8 2017·1:21 PM ES

Ryan Browne
@RYAN_BROWNE_



devops199 @devops199
will i get arrested for this? 😟
0x642483b7936b505dbe2e735cc140f29ddfddb3f3e39efa549707d98e0e0b18421b
0xae7168deb525862f4fee37d987a971b385b96952

Tienus @Tienus
@devops199 you are the one that called the kill tx?

devops199 @devops199
yes
i'm eth newbie..just learning

qx133 @qx133
you are famous now haha

devops199 @devops199
sending kill() destroy() to random contracts
you can see my history
😟((((((((((((((((((((((((((((((

Xavier @n3xco
can't make an omelet without breaking some eggs
i guess

https://github.com/openethereum/parity-ethereum/issues/6995

Bugs: Discussion

What would happen if an inflation bug was exploited in Bitcoin in 2025?

Which sorts of bugs are the most dangerous? netsplits? inflation?

What countermeasures can we take to reduce the risk and the impact of a serious software vulnerability exploited in the wild against cryptocurrencies?

Is a vulnerability that works against 100% of nodes worse or better than one that exploits 30% of the nodes?

Cryptology* assumptions

Hash functions:

- Collision Resistance
- Preimage Resistance

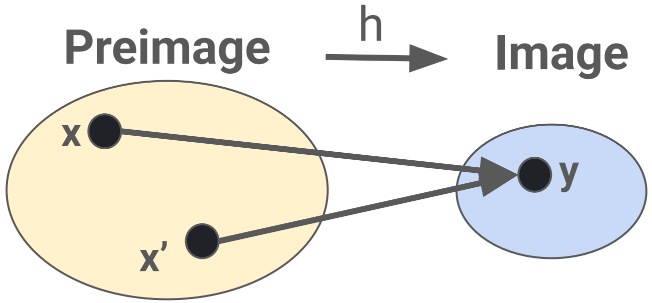
Signatures:

- Forgery

Giechaskiel et. al. When the “Crypto” in Cryptocurrencies Breaks: Bitcoin Security Under Broken Primitives
https://ilias.giechaskiel.com/papers/2018_1_bitcoin_sp.pdf

Hash functions: collisions

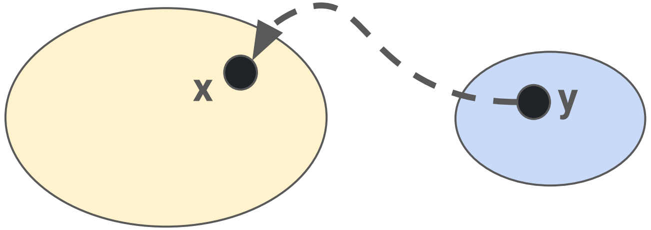
Collision Resistance: no one can compute
…two diffs. inputs that result in the same output
Find x, x’ **s.t.** x != x’ AND h(x)=h(x’)



Can cause netsplits…

Hash functions: Preimages

Preimage Resistance: no one can compute
…an input (preimage) x from just the output y
Given y find x **s.t.** h(x) = y



P2SH
Output ← H(redeem script)

If the preimage is a random value, will it be a script the attacker can spend?
What is the probability that a random 256 bytes is a valid redeem script?

Attacker might or might not have some control over the preimage

Hash functions

While the consequences would be bad, we are likely to get advance warning and there are easy fixes for collisions.

For collisions:

Soft fork in a second hash function algorithm

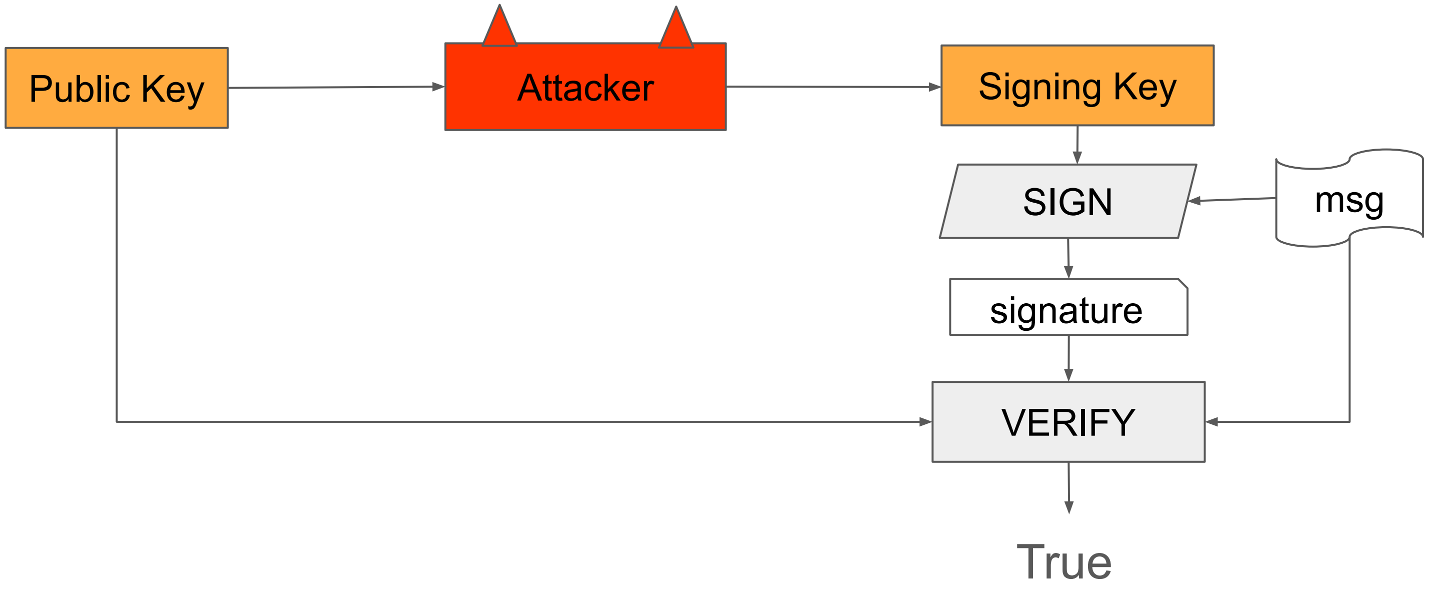
For preimages:

it is more complex since it depends on the specifics of the attack i.e. how much control the attacker has.

We will come back to what to do about a very bad preimage attack

Signature forgeries

Let's say we have an attacker who after seeing a public key can compute the associated signing key?



Signature forgeries

Bitcoin and cryptocurrencies in general can be reduced to the following primal goal:
Authenticate ownership of coins

If the signature scheme that enables ownership authn. is broken, anyone can claim anyone else's coins. Does ownership still exist?

Could Bitcoin survive a vuln that breaks the authn. of ownership

Signature forgeries

Imagine that all of Bitcoin was P2PK (Pay-to-Public-Key)

If we get a 1 year warning, what could we do?

Introduce a new signature scheme, get people to move coins

What about the people that don’t move? Lost coins?

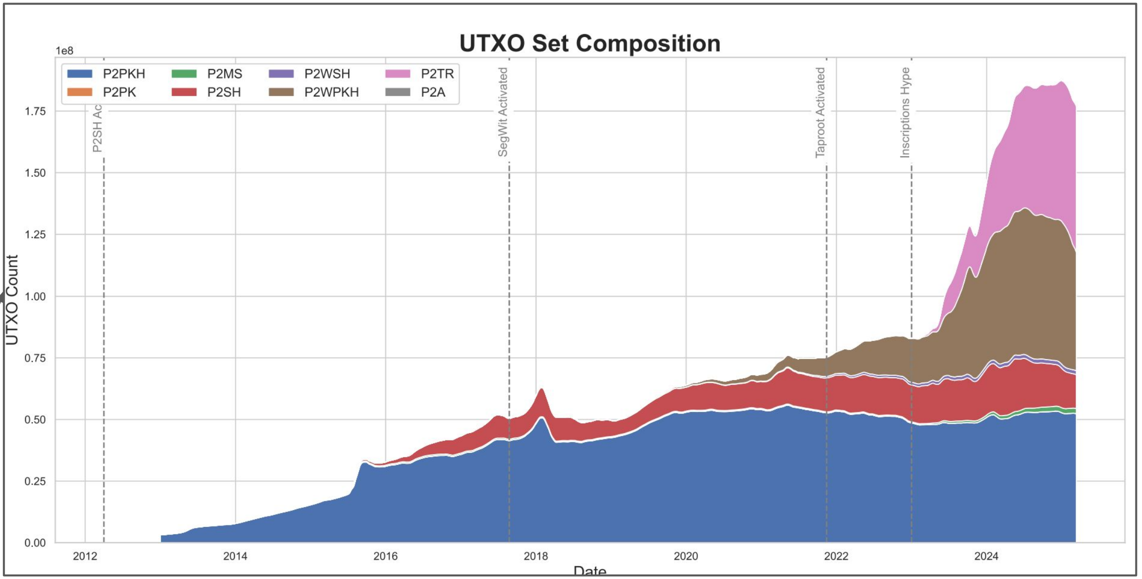
If we get no warning, what could we do?

Signature forgeries

Thankfully not all of Bitcoin was P2PK (Pay-to-Public-Key)

P2PKH, P2WPKH,
P2SH, P2WSH
… hide the public key
behind a hash.

# of outputs,
not # of coins



Bitcoin and Quantum Computing: Current Status and Future Directions (may 2025)

Signature forgeries

This creates two classes of output vulnerabilities:

- **Long exposure** - public key has been exposed to attackers either because of address reuse or because the public key revealed on-chain

- **Short exposure** - public key is not yet known to attackers, only exposed at the moment of spending. Attacker race to recover signing key before txn is confirmed.

	Public key revealed	**Window of Vuln.**
Long exposure	At output creation	Unlimited
Short exposure	At output spend	Before txn confirms

Return of the Preimage attacks

This creates two classes of output vulnerabilities:

- **Long exposure** - public key has been exposed to attackers either because of address reuse or because the public key revealed on-chain

- **Short exposure** - public key is not yet known to attackers, only exposed at the moment of spending. Attacker race to recover signing key before txn is confirmed.

A preimage attack in which an attacker can choose the preimage is equivalent to a **long exposure** vulnerability.

Such high control preimage attacks are extremely unlikely, if you are curious ask me why at the end of class?

Signature forgeries: Quantum

Classical computers - the computers we have today.

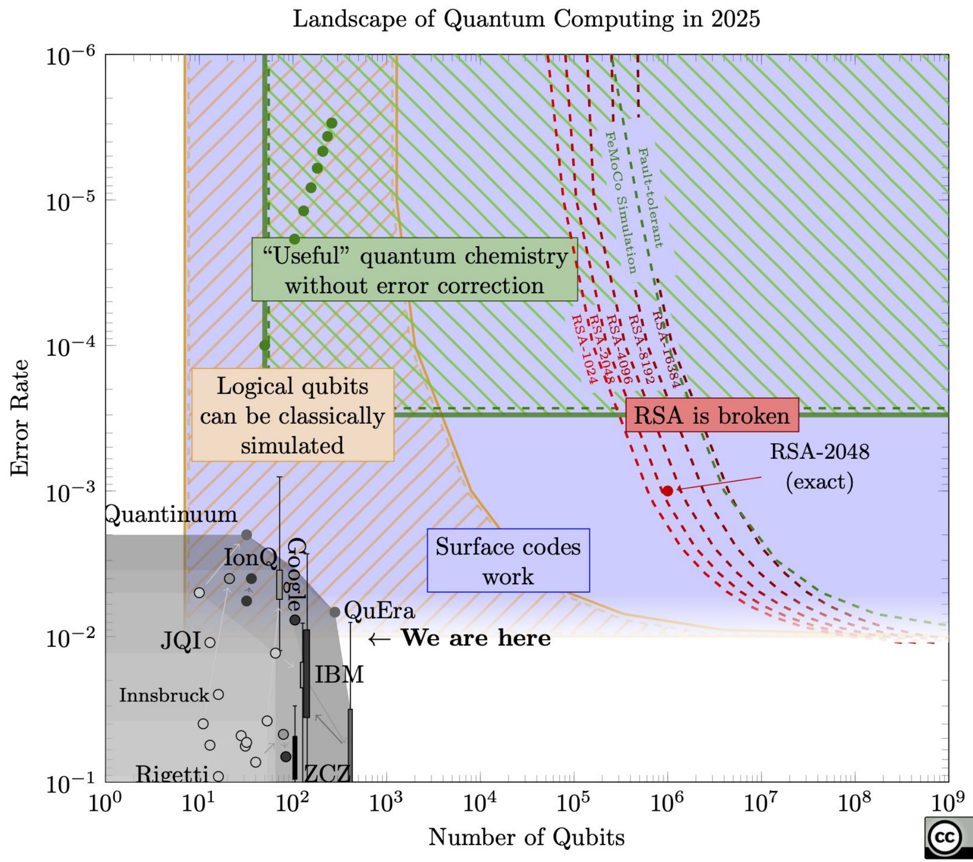
Quantum Computers (QC) use quantum mechanics to provide higher performance than classical computers for a small number of specific problems.

QC do not break most cryptography. In many cases QC are likely to be slower at breaking cryptography than classical computers.

…Unfortunately one the specific problems that QC are better at is breaking particular signature algorithms used by most cryptocurrencies (including BTC).

Thus, at some point in the future QCs may enable signature forgery attacks.

Quantum Computers (QC) are not cryptologically relevant yet



We need to ~13 doublings to break RSA while shrinking error rate.

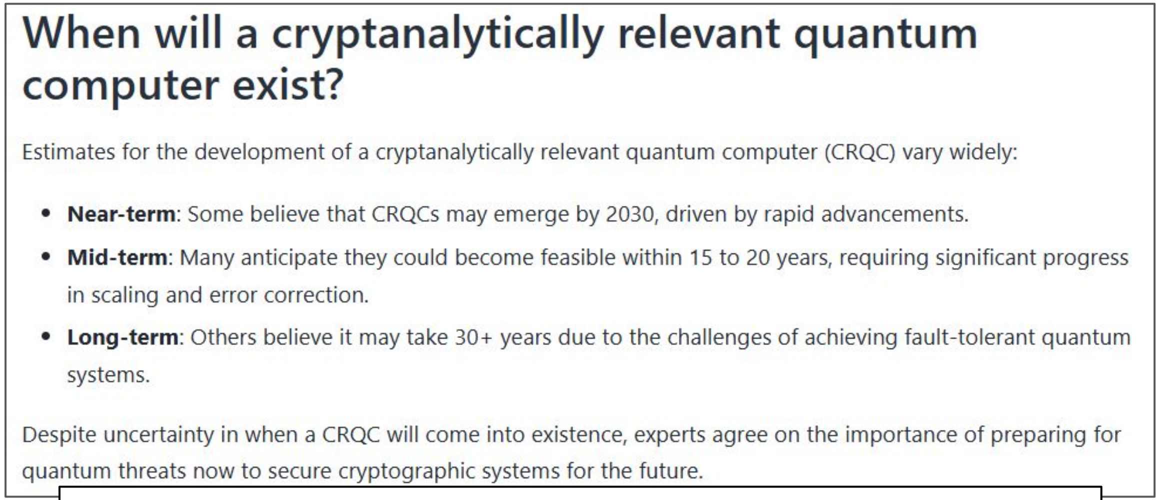
Note:
Both of the axis are log scales

[Jaques's Landscape of Quantum Computing (2025)]

Signature forgeries: Quantum

The best publicly known Quantum Computers (QC) today are not cryptologically relevant

Estimates are
5 years
to
30+ years



When will a cryptanalytically relevant quantum computer exist?

Estimates for the development of a cryptanalytically relevant quantum computer (CRQC) vary widely:

- **Near-term**: Some believe that CRQCs may emerge by 2030, driven by rapid advancements.
- **Mid-term**: Many anticipate they could become feasible within 15 to 20 years, requiring significant progress in scaling and error correction.
- **Long-term**: Others believe it may take 30+ years due to the challenges of achieving fault-tolerant quantum systems.

Despite uncertainty in when a CRQC will come into existence, experts agree on the importance of preparing for quantum threats now to secure cryptographic systems for the future.

National Institute of Standards and Technology (NIST) Quantum FAQ

Signature forgeries: Quantum

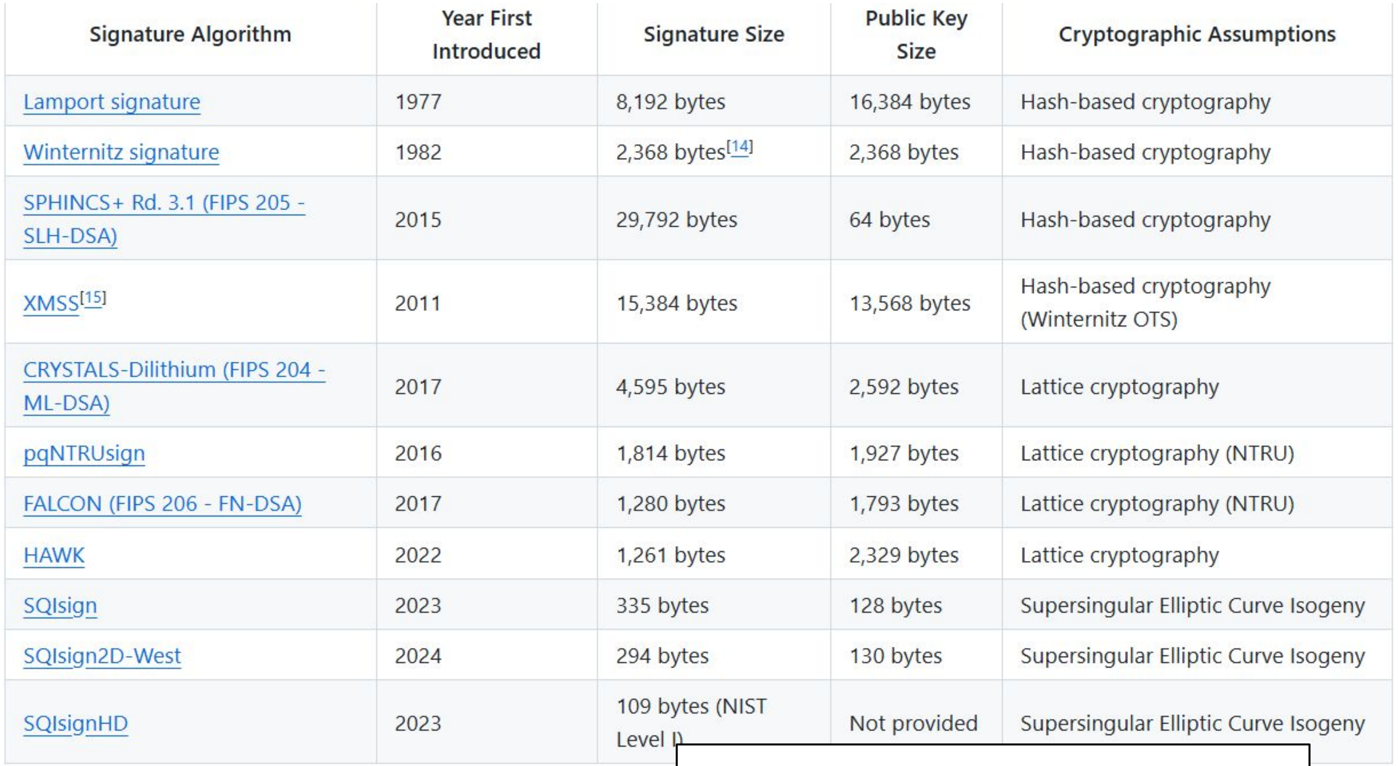
How to mitigate risks QCs breaking signature algorithms?

Treat it the same way as any other cryptologic break in a signature alg:
switch to a new algorithm.

Cryptography community has developed Post-Quantum signature algorithms secure against known attacks.

Simple right?

PQ Signatures are big!



Signature Algorithm	Year First Introduced	Signature Size	Public Key Size	Cryptographic Assumptions
Lamport signature	1977	8,192 bytes	16,384 bytes	Hash-based cryptography
Winternitz signature	1982	2,368 bytes[14]	2,368 bytes	Hash-based cryptography
SPHINCS+ Rd. 3.1 (FIPS 205 - SLH-DSA)	2015	29,792 bytes	64 bytes	Hash-based cryptography
XMSS[15]	2011	15,384 bytes	13,568 bytes	Hash-based cryptography (Winternitz OTS)
CRYSTALS-Dilithium (FIPS 204 - ML-DSA)	2017	4,595 bytes	2,592 bytes	Lattice cryptography
pqNTRUsign	2016	1,814 bytes	1,927 bytes	Lattice cryptography (NTRU)
FALCON (FIPS 206 - FN-DSA)	2017	1,280 bytes	1,793 bytes	Lattice cryptography (NTRU)
HAWK	2022	1,261 bytes	2,329 bytes	Lattice cryptography
SQIsign	2023	335 bytes	128 bytes	Supersingular Elliptic Curve Isogeny
SQIsign2D-West	2024	294 bytes	130 bytes	Supersingular Elliptic Curve Isogeny
SQIsignHD	2023	109 bytes (NIST Level I)	Not provided	Supersingular Elliptic Curve Isogeny

BIP-360 Pay to Quantum Resistant Hash

Signature forgeries: Quantum

Approaches to mitigate risk signature forgeries (quantum or otherwise):

1. What alternative algorithms do you use and
   What if they are slower or bigger?
2. How do you get people to switch to using the new algorithms in time?
3. What do you do about outputs that don't switch?

Let's discuss

Conclusion

Cryptocurrencies are pretty resilient to most bugs and vulnerabilities.

Biggest danger is no longer being able to authenticate ownership of coins (break in signature algorithm).

Things that can break:

- Consensus/protocol assumptions
- Incentives
- Software
    - Bugs in wallets
    - Bugs in nodes
    - Bugs in protocols
- Communication assumptions i.e. partitions
- Cryptographic assumptions
    - Hash function breaks
    - Signature algorithms breaks

Reminders & Next Week & Questions

- Reminder:
  - Sign up for the class Discord
  - Source of truth https://github.com/mit-dci/cde-2025
- Questions?

The End

Imagining a worst case non-cryptologic exploit

Bitcoin relies on its underlying cryptography so an attacker that could
1. instantly forge ECDSA signatures,
2. break random number generators,
3. or generate arbitrary SHA256 preimages

...would be very very bad.

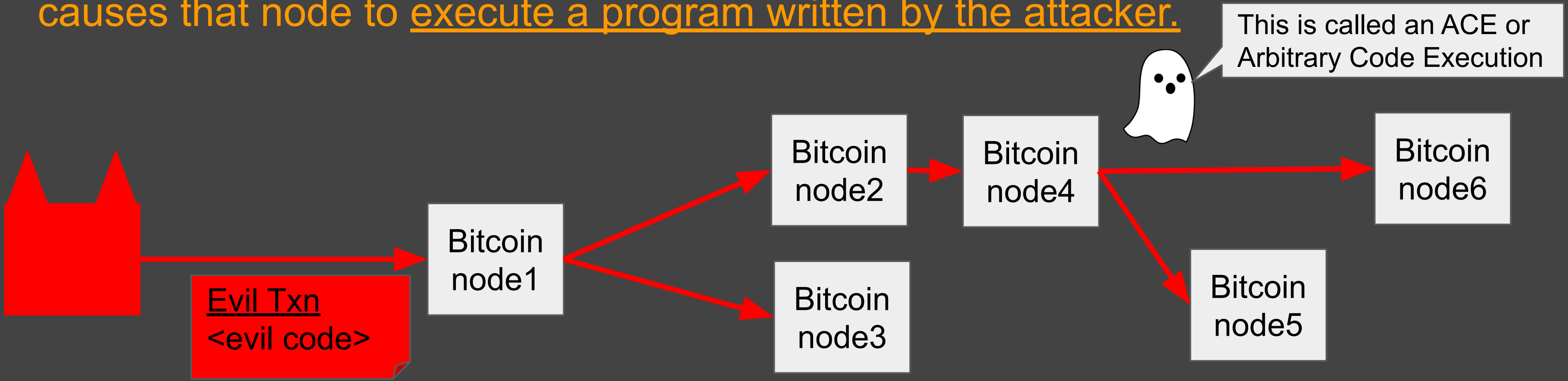
However, sudden breaks in standardized cryptography are rare **whereas** software vulnerabilities are common.

Let’s think about the worst possible software vulnerability in Bitcoin-core.

This experiment is intended as a starting point for discussion not a conclusion. Disagreement encouraged!!!

Imagining a worst case non-cryptologic Exploit (a worm)

Consider a valid transaction which when evaluated for validity by a Bitcoin node causes that node to execute a program written by the attacker.



1. Attacker sends Evil_Txn,
2. Node evaluates Evil_Txn,
   ...triggers the vuln
   ...and runs the evil code

3. As it is a valid txn node1 sends
   ...Evil_Txn to node2 and node3.
4. It gets gossiped across the network
   ...each time running the evil code

Given this capability what is the worst thing an attacker can do?

Imagining a worst case non-cryptologic Exploit

A statement: Attacks on Bitcoin which result in
(1). a short-chain
(2). that almost everyone agrees should be invalid
are reversible.

This is not a fact, but a conjecture. There are likely many exceptions and good arguments not to believe this.

Attacks that keep the blockchain valid:

- Filter/distort what a node tells the user about the blockchain:
  hide evidence of other attacks,
  and make certain invalid txns appear valid and confirmed.
- Leak keys, compromise random number generation, replace pubkeys

etc...

Imagining a worst case non-cryptologic Exploit

Such an exploit would be very difficult to develop as it would need to work (1). on different versions of Bitcoin-core, (2). on different OSes, (3). on different architectures and (4). still be valid transaction.

More likely would be an exploit that only targets a particular OS and Bitcoin-core version.

Evil code might be able to patch Bitcoin-core to treat Evil_Txn as valid.

I am not aware of any vulnerabilities in Bitcoin that allowed Arbitrary Code Execution (ACE) like this.

While this would be bad, it might be a very unlikely threat