low budget
segwit safari
residency 2019
basics
- malleability fix
- malleability fix
  - attacks?
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- no quadratic sighash
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- no quadratic sighash
- better p2sh security
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- no quadratic sighash
- better p2sh security
  - HASH160 <20 bytes> EQUAL vs OP_0 <32 bytes>
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- no quadratic sighash
- better p2sh security
  - HASH160 <20 bytes> EQUAL vs OP_0 <32 bytes>
- script upgradeability
- malleability fix
  - attacks?
  - nVersion|txins|txouts|nLockTime
  - nVersion|marker|flag|txins|txouts|witness|nLockTime
- no quadratic sighash
- better p2sh security
  - HASH160 <20 bytes> EQUAL vs OP_0 <32 bytes>
- script upgradeability
- block size increase
  - 4MB cap in theory, 1.6-2MB in practice
witness commitment in coinbase
```cpp
std::vector<unsigned char> GenerateCoinbaseCommitment(CBlock& block, const CBlockIndex* pindexPrev, const Consensus::Params& consensusParams)
{
    std::vector<unsigned char> commitment;
    int commitpos = GetWitnessCommitmentIndex(block);
    std::vector<unsigned char> ret(32, 0x00);
    if (consensusParams.vDeployments[Consensus::DEPLOYMENT_SEGWIT].nTimeout != 0) {
        if (commitpos == -1) {
            uint256 witnessroot = BlockWitnessMerkleRoot(block, nullptr);
            CHash256().Write(witnessroot.begin(), 32).Write(ret.data(), 32).Finalize(witnessroot.begin());
            COut out; out.nValue = 0;
            out.scriptPubKey.resize(38);
            out.scriptPubKey[0] = OP_RETURN;
            out.scriptPubKey[1] = 0x24;
            out.scriptPubKey[2] = 0xaa;
            out.scriptPubKey[3] = 0x21;
            out.scriptPubKey[4] = 0xa9;
            out.scriptPubKey[5] = 0xed;
            memcpy(&out.scriptPubKey[6], witnessroot.begin(), 32);
            commitment = std::vector<unsigned char>(out.scriptPubKey.begin(), out.scriptPubKey.end());
            CMutableTransaction tx(*out.scriptPubKey.begin(), out.scriptPubKey.end());
            tx.vout.push_back(out);
            block.vtx[0] = MakeTransactionRef(std::move(tx));
        }
    }
    UpdateUncommittedBlockStructures(block, pindexPrev, consensusParams);
    return commitment;
}
```
```cpp
uint256 BlockWitnessMerkleRoot(const CBlock& block, bool* mutated) {
    std::vector<uint256> leaves;
    leaves.resize(block.vtx.size());
    leaves[0].SetNull(); // The witness hash of the coinbase is 0.
    for (size_t s = 1; s < block.vtx.size(); s++) {
        leaves[s] = block.vtx[s]->GetWitnessHash();
    }
    return ComputeMerkleRoot(std::move(leaves), mutated);
}
```
basics

witness programs
p2wpkh

witness:  
<signature>  <pubkey>
scriptSig:  (empty)
scriptPubKey:  0  <20-byte-key-hash>
              (0x0014{20-byte-key-hash})
p2wsh

witness: 0 <signature1> <1 <pubkey1> <pubkey2> 2 CHECKMULTISIG>
scriptSig: (empty)
scriptPubKey: 0 <32-byte-hash>
(0x0020{32-byte-hash})
activation

bip9
from bip9 (https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki)
activation

bip148
// Check if Segregated Witness is Locked In
bool IsWitnessLockedIn(const CBlockIndex* pindexPrev, const Consensus::Params& params)
{
    LOCK(cs_main);
    return (VersionBitsState(pindexPrev, params, Consensus::DEPLOYMENT_SEGWIT,
        versionbitscache) == THRESHOLD_LOCKED_IN);
}

// BIP148 mandatory segwit signalling.
int64_t nMedianTimePast = pindex->GetMedianTimePast();
if ( (nMedianTimePast >= 1501545600) &&  // Tue 01 Aug 2017 00:00:00 UTC
    (nMedianTimePast <= 1510704000) &&  // Wed 15 Nov 2017 00:00:00 UTC
    (!IsWitnessLockedIn(pindex->pprev, chainparams.GetConsensus()) &&  // Segwit is not
     !IsWitnessEnabled(pindex->pprev, chainparams.GetConsensus())) )   // and is not active.
{
    bool fVersionBits = (pindex->nVersion & VERSIONBITS_TOP_MASK) == VERSIONBITS_TOP_BITS;
    bool fSegbit = (pindex->nVersion & VersionBitsMask(chainparams.GetConsensus(),
        Consensus::DEPLOYMENT_SEGWIT)) != 0;
    if (!(fVersionBits && fSegbit)) {
        return state.DoS(0, error("ConnectBlock(): relayed block must signal for segwit,
        please upgrade"), REJECT_INVALID, "bad-no-segwit");
    }
}
"BIP 148 would introduce a new consensus rule that softforks out non-segwit signalling blocks in some time period. I reject this consensus rule as both arbitrary and needlessly disruptive. Bitcoin's primary purpose is to reach consensus on the state of a shared ledger, and even though I think the Bitcoin network ought to adopt segwit, I don't think that concern trumps the goal of not splitting the network."

sdaftuar

https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-May/014377.html
activation

bip8
activation

bip91
(c)overt asicboost
from jeremy rubin (http://www.mit.edu/~jlrubin//public/pdfs/Asicboost.pdf)
peer relations
new serialization format (bip144)

<table>
<thead>
<tr>
<th>Field Size</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>version</td>
<td>int32_t</td>
<td>Transaction data format version</td>
</tr>
<tr>
<td>1</td>
<td>marker</td>
<td>char</td>
<td>Must be zero</td>
</tr>
<tr>
<td>1</td>
<td>flag</td>
<td>char</td>
<td>Must be nonzero</td>
</tr>
<tr>
<td>1+</td>
<td>txin_count</td>
<td>var_int</td>
<td>Number of transaction inputs</td>
</tr>
<tr>
<td>41+</td>
<td>txins</td>
<td>txin[]</td>
<td>A list of one or more transaction inputs</td>
</tr>
<tr>
<td>1+</td>
<td>txout_count</td>
<td>var_int</td>
<td>Number of transaction outputs</td>
</tr>
<tr>
<td>9+</td>
<td>txouts</td>
<td>txouts[]</td>
<td>A list of one or more transaction outputs</td>
</tr>
<tr>
<td>1+</td>
<td>script_witnesses</td>
<td>script_witnesses[]</td>
<td>The witness structure as a serialized byte array</td>
</tr>
<tr>
<td>4</td>
<td>lock_time</td>
<td>uint32_t</td>
<td>The block number or timestamp until which the transaction is locked</td>
</tr>
</tbody>
</table>

Drivers supporting this BIP will be able to dialog wih between the old serialization format (without the witness) and this new format.

https://github.com/bitcoin/bips/blob/master/bip-0144.mediawiki
**a note on hashes**

“Transaction hashes used in the transaction merkle tree and txin outpoints are always computed using the old non-witness serialization.”

Somewhat deceptive - witness txs don’t include txin scriptSig (signature) data

peer relations

new messages (getdata):
- MSG_WITNESS_TX
- MSG_WITNESS_BLOCK
peer relations
git grep NODE_WITNESS
connecting to relevant services (then)

| 1679 | + | // only consider nodes missing relevant services after 40 failed attempts |
| 1680 | + | if ((addr.nServices & nRelevantServices) != nRelevantServices && nTries < 40) |
| 1681 | + | continue; |
| 1682 | + | |

https://github.com/bitcoin/bitcoin/pull/8149/commits/b8a97498df1e83f8dcc49bc3fa4344f9e9799242#diff-9a82240fe7dfe86564178691cc57f2f1R1679
connecting to relevant services

```cpp
// only consider very recently tried nodes after 30 failed attempts
if (nANow - addr.nLastTry < 600 && nTries < 30)
    continue;

// for non-feelers, require all the services we'll want,
// for feelers, only require they be a full node (only because most
// SPV clients don't have a good address DB available)
if (!fFeeler && !HasAllDesirableServiceFlags(addr.nServices)) {
    continue;
} else if (fFeeler && !MayHaveUsefulAddressDB(addr.nServices)) {
    continue;
}

// do not allow non-default ports, unless after 50 invalid addresses selected already
if (addr.GetPort() != Params().GetDefaultPort() && nTries < 50)
    continue;
```
Replace relevant services logic with a function suite.

Adds `HasAllRelevantServices` and `GetRelevantServices`, which check for `NETWORK|WITNESS`.

This changes the following:
* Removes `nRelevantServices` from `CConnman`, disconnecting it a bit more from protocol-level logic.
* Replaces our sometimes-connect-to-!WITNESS-nodes logic with simply always requiring `WITNESS|NETWORK` for outbound non-feeler connections (feeler still only require `NETWORK`).
* This has the added benefit of removing `nServicesExpected` from `CNode` - instead letting net processing's `VERSION` message handling simply check `HasAllRelevantServices`.
* This implies we believe `WITNESS` nodes to continue to be a significant majority of nodes on the network, but also because we cannot sync properly from !WITNESS nodes, it is strange to continue using our valuable outbound slots on them.
if (chainparams.GetConsensus().vDeployments[Consensus::DEPLOYMENT_SEGWIT].nTimeout != 0) {
    // Only advertise witness capabilities if they have a reasonable start time.
    // This allows us to have the code merged without a defined softfork, by setting its
    // end time to 0.
    // Note that setting NODE_WITNESS is never required; the only downside from not
    // doing so is that after activation, no upgraded nodes will fetch from you.
    nLocalServices = ServiceFlags(nLocalServices | NODE_WITNESS);
}
// check against previous transactions
// This is done last to help prevent CPU exhaustion denial-of-service attacks.

if (!CheckInputs(tx, state, view, true, STANDARD_SCRIPT_VERIFY_FLAGS, true))
    return false; // state filled in by CheckInputs

if (!CheckInputs(tx, state, view, true, STANDARD_SCRIPT_VERIFY_FLAGS, true)) {
    // SCRIPT_VERIFY_CLEANSTACK requires SCRIPT_VERIFY_WITNESS, so we
    // need to turn both off, and compare against just turning off CLEANSTACK
    // to see if the failure is specifically due to witness validation.
    if (CheckInputs(tx, state, view, true, STANDARD_SCRIPT_VERIFY_FLAGS & ~(SCRIPT_VERIFY_WITNESS | SCRIPT_VERIFY_CLEANSTACK), true)) {
        // Only the witness is wrong, so the transaction itself may be fine.
        state.SetCorruptionPossible();
    }
    return false;
}
/**
 * If we've announced NODE_WITNESS to this peer: whether the peer sends witnesses in cmpctblocks/blocktxns,
 * otherwise: whether this peer sends non-witnesses in cmpctblocks/blocktxns.
 */

bool fSupportsDesiredCmpctVersion;
extensibility
extensibility

script versioning
```cpp
1269  +   }
1270  +   } else if (flags & SCRIPT_VERIFY_DISCOURAGE_UPGRADABLE_WITNESS_PROGRAM) {
1271  +       return set_error(serror, SCRIPT_ERR_DISCOURAGE_UPGRADABLE_WITNESS_PROGRAM);
1272  +   } else {
1273  +     // Higher version witness scripts return true for future softfork compatibility
1274  +     return set_success(serror);
1275  +   }
1276  +
```

https://github.com/bitcoin/bitcoin/pull/8149/commits/449f9b8debcceb61a92043bc7031528a53627c47#diff-be2905e2f5218ecdbe4e55637dac75f3R1273
extensibility

unused coinbase commitment
```c++
+ void UpdateUncommittedBlockStructures(CBlock& block, const CBlockIndex* pindexPrev, const Consensus::Params& consensusParams);
+ {
+   int commitpos = GetWitnessCommitmentIndex(block);
+   static const std::vector<unsigned char> nonce(32, 0x00);
+   if (commitpos != -1 && IsWitnessEnabled(pindexPrev, consensusParams) && block.vtx[0].wit.IsEmpty()) {
+     block.vtx[0].wit.vtxinwit.resize(1);
+     block.vtx[0].wit.vtxinwit[0].scriptWitness.stack.resize(1);
+     block.vtx[0].wit.vtxinwit[0].scriptWitness.stack[0] = nonce;
+   }
+ }
```
the worst line in segwit
cmon man that's 354 characters
j/k pieter thanks for segwit

but for real
maybe let’s do 120col
questions
questions
was segwit the right change?
questions

what’s involved in schnorr/taproot upgrade?
questions

what’s a likely deployment mechanism?
- segwit PR (rebased):
  https://github.com/bitcoin/bitcoin/pull/8149
- Peter Todd’s code review:
  https://petertodd.org/2016/segwit-consensus-critical-code-review
- BIPs 141-144: you’ll read ‘em multiple times
- test/functional/test_framework/messages.py: quick ref for message formats
it's my 30th birthday
come to Madison Sq Park
have some red wine
make unqualified statements
about consensus critical code