Popchain mainnet audit report

Yong-hyu Ban

July 25, 2019

1 Target of analysis

- Full project except for BerkeleyDB and libboost component

2 Analysis method

- Full project static analysis using Synopsis® Coverity® Scan
- Manual code inspection

3 Analysis result

3.1 Static analysis using Synopsis® Coverity® Scan - High impact issues

3.1.1 CID 242525 - Pointer to local outside scope (use-after-free)

In `src/main.cpp:AddInvalidSpendsToMap`, reference to 'publicSpend' local variable is used after 'publicSpend' is freed from the stack. It can be lead to data corruption.

```
2473 void AddInvalidSpendsToMap(const CBlock& block)
2474 {
2475     for (const CTransaction& tx : block.vtx) {
2476         if (!tx.ContainsZerocoins())
2477             continue;
2478 
2479         //Check all zerocoinspends for bad serials
2480         for (const CTxIn& in : tx.vin) {
2481             bool isPublicSpend = in.IsZerocoinPublicSpend();
2482             if (in.IsZerocoinSpend() || isPublicSpend) {
2483 
2484                 CoinSpend* spend;
2485                 if (isPublicSpend) {
2486                     libzerocoin::ZerocoinParams* params = Params().Zerocoin_Params(false);
2487                     PublicCoinSpend publicSpend(params);
2488                     CValidationState state;
2489                     if (!ZPCHModule::ParseZerocoinPublicSpend(in, tx, state, publicSpend)){
2490                         throw runtime_error("Failed to parse public spend");
2491                     }
2492                     spend = &publicSpend;
2493                 } else {
2494                     CoinSpend spendObj = TxInToZerocoinSpend(in);
2495                     spend = &spendObj;
2496                 }
2497 
2498                 //If serial is not valid, mark all outputs as bad
2499                 if (!spend->HasValidSerial(Params().Zerocoin_Params(false))) {
2500                     mapInvalidSerials[spend->getCoinSerialNumber()] = spend->getDenomination() * COIN;
```

3.1.2 CID 242511 - Resource leak

In `src/wallet/wallet.cpp:CWallet::AutoCombineDust`, 'coinControl' is dynamically allocated but not freed when coinControl->HasSelected() is false. It may lead to a memory leak which can disrupt stability.

```
3388 void CWallet::AutoCombineDust()
3389 {
3390     LOCK2(cs_main, cs_wallet);
3391     if (chainActive.Tip()->nTime < (GetAdjustedTime() - 300) || IsLocked()) {
3392         return;
3393     }
3394
3395     map<CBitcoinAddress, vector<COutput> > mapCoinsByAddress = AvailableCoinsByAddress(true,
          nAutoCombineThreshold * COIN);
3396
3397     //coins are sectioned by address. This combination code only wants to combine inputs that belong to
          the same address
3398     for (map<CBitcoinAddress, vector<COutput> >::iterator it = mapCoinsByAddress.begin(); it !=
          mapCoinsByAddress.end(); it++) {
3399         vector<COutput> vCoins, vRewardCoins;
3400         bool maxSize = false;
3401         vCoins = it->second;
3402
3403         // We don't want the tx to be refused for being too large
3404         // we use 50 bytes as a base tx size (2 output: 2*34 + overhead: 10 -> 90 to be certain)
3405         unsigned int txSizeEstimate = 90;
3406
3407         //find masternode rewards that need to be combined
3408         CCoinControl* coinControl = new CCoinControl();
3409         CAmount nTotalRewardsValue = 0;
3410         for (const COutput& out : vCoins) {
3411             if (!out.fSpendable)
3412                 continue;
3413             //no coins should get this far if they dont have proper maturity, this is double checking
3414             if (out.tx->IsCoinStake() && out.tx->GetDepthInMainChain() < Params().COINBASE_MATURITY() +
                  1)
3415                 continue;
3416
3417             COutPoint outpt(out.tx->GetHash(), out.i);
3418             coinControl->Select(outpt);
3419             vRewardCoins.push_back(out);
3420             nTotalRewardsValue += out.Value();
3421
3422             // Combine to the threshold and not way above
3423             if (nTotalRewardsValue > nAutoCombineThreshold * COIN)
3424                 break;
3425
3426             // Around 180 bytes per input. We use 190 to be certain
3427             txSizeEstimate += 190;
3428             if (txSizeEstimate >= MAX_STANDARD_TX_SIZE - 200) {
3429                 maxSize = true;
3430                 break;
3431             }
3432         }
3433
3434         //if no inputs found then return
3435         if (!coinControl->HasSelected())
3436             continue;
```

3.1.3 CID 242479 - Pointer to local outside scope (use-after-free)

In `src/miner.cpp:CreateNewBlock`, reference to 'publicSpoend' and 'spendObj' member variable is used after 'publicSpend' and 'spendObj' is freed from the stack. It can be lead to data corruption.

```
CBlockTemplate* CreateNewBlock(const CScript& scriptPubKeyIn, CWallet* pwallet, bool fProofOfStake)
{
    CReserveKey reservekey(pwallet);

    // Create new block
    unique_ptr<CBlockTemplate> pblocktemplate(new CBlockTemplate());
    if (!pblocktemplate.get())
        return NULL;
    CBlock* pblock = &pblocktemplate->block; // pointer for convenience

    // Tip
    CBlockIndex* pindexPrev = nullptr;
    {   // Don't keep cs_main locked
        LOCK(cs_main);
        pindexPrev = chainActive.Tip();
    }

    const int nHeight = pindexPrev->nHeight + 1;

    // Make sure to create the correct block version after zerocoin is enabled
    bool fZerocoinActive = nHeight >= Params().Zerocoin_StartHeight();
    pblock->nVersion = fZerocoinActive ? 5 : 3;

    // -regtest only: allow overriding block.nVersion with
    // -blockversion=N to test forking scenarios
    if (Params().MineBlocksOnDemand()) {
        if (fZerocoinActive)
            pblock->nVersion = 5;
        else
            pblock->nVersion = 3;

        pblock->nVersion = GetArg("-blockversion", pblock->nVersion);
    }

    // Create coinbase tx
    CMutableTransaction txNew;
    txNew.vin.resize(1);
    txNew.vin[0].prevout.SetNull();
    txNew.vout.resize(1);
    txNew.vout[0].scriptPubKey = scriptPubKeyIn;
    pblock->vtx.push_back(txNew);
    pblocktemplate->vTxFees.push_back(-1);   // updated at end
    pblocktemplate->vTxSigOps.push_back(-1); // updated at end

    // ppcoin: if coinstake available add coinstake tx
    static int64_t nLastCoinStakeSearchTime = GetAdjustedTime(); // only initialized at startup

    if (fProofOfStake) {
        boost::this_thread::interruption_point();
        pblock->nTime = GetAdjustedTime();
        pblock->nBits = GetNextWorkRequired(pindexPrev, pblock);
        CMutableTransaction txCoinStake;
        int64_t nSearchTime = pblock->nTime; // search to current time
        bool fStakeFound = false;
        if (nSearchTime >= nLastCoinStakeSearchTime) {
            unsigned int nTxNewTime = 0;
            if (pwallet->CreateCoinStake(*pwallet, pblock->nBits, nSearchTime - nLastCoinStakeSearchTime,
                  txCoinStake, nTxNewTime)) {
                pblock->nTime = nTxNewTime;
                pblock->vtx[0].vout[0].SetEmpty();
                pblock->vtx.push_back(CTransaction(txCoinStake));
                fStakeFound = true;
```

```
            }
            nLastCoinStakeSearchInterval = nSearchTime - nLastCoinStakeSearchTime;
            nLastCoinStakeSearchTime = nSearchTime;
        }

        if (!fStakeFound) {
            LogPrint("staking", "CreateNewBlock(): stake not found\n");
            return NULL;
        }
    }

    // Largest block you're willing to create:
    unsigned int nBlockMaxSize = GetArg("-blockmaxsize", DEFAULT_BLOCK_MAX_SIZE);
    // Limit to betweeen 1K and MAX_BLOCK_SIZE-1K for sanity:
    unsigned int nBlockMaxSizeNetwork = MAX_BLOCK_SIZE_CURRENT;
    nBlockMaxSize = std::max((unsigned int)1000, std::min((nBlockMaxSizeNetwork - 1000), nBlockMaxSize));

    // How much of the block should be dedicated to high-priority transactions,
    // included regardless of the fees they pay
    unsigned int nBlockPrioritySize = GetArg("-blockprioritysize", DEFAULT_BLOCK_PRIORITY_SIZE);
    nBlockPrioritySize = std::min(nBlockMaxSize, nBlockPrioritySize);

    // Minimum block size you want to create; block will be filled with free transactions
    // until there are no more or the block reaches this size:
    unsigned int nBlockMinSize = GetArg("-blockminsize", DEFAULT_BLOCK_MIN_SIZE);
    nBlockMinSize = std::min(nBlockMaxSize, nBlockMinSize);

    // Collect memory pool transactions into the block
    CAmount nFees = 0;

    {
        LOCK2(cs_main, mempool.cs);

        CBlockIndex* pindexPrev = chainActive.Tip();
        const int nHeight = pindexPrev->nHeight + 1;
        CCoinsViewCache view(pcoinsTip);

        // Priority order to process transactions
        list<COrphan> vOrphan; // list memory doesn't move
        map<uint256, vector<COrphan*> > mapDependers;
        bool fPrintPriority = GetBoolArg("-printpriority", false);

        // This vector will be sorted into a priority queue:
        vector<TxPriority> vecPriority;
        vecPriority.reserve(mempool.mapTx.size());
        for (map<uint256, CTxMemPoolEntry>::iterator mi = mempool.mapTx.begin();
             mi != mempool.mapTx.end(); ++mi) {
            const CTransaction& tx = mi->second.GetTx();
            if (tx.IsCoinBase() || tx.IsCoinStake() || !IsFinalTx(tx, nHeight)){
                continue;
            }
            if(GetAdjustedTime() > GetSporkValue(SPORK_16_ZEROCOIN_MAINTENANCE_MODE) && tx.
                 ContainsZerocoins()){
                continue;
            }

            COrphan* porphan = NULL;
            double dPriority = 0;
            CAmount nTotalIn = 0;
            bool fMissingInputs = false;
            uint256 txid = tx.GetHash();
            bool hasZerocoinSpends = tx.HasZerocoinSpendInputs();
```

```
223             if (hasZerocoinSpends)
224                 nTotalIn = tx.GetZerocoinSpent();
225
226             for (const CTxIn& txin : tx.vin) {
227                 //zerocoinspend has special vin
228                 if (hasZerocoinSpends) {
229                     //Give a high priority to zerocoinspends to get into the next block
230                     //Priority = (age^6+100000)*amount - gives higher priority to zpchs that have been in
                                mempool long
231                     //and higher priority to zpchs that are large in value
232                     int64_t nTimeSeen = GetAdjustedTime();
233                     double nConfs = 100000;
234
235                     auto it = mapZerocoinspends.find(txid);
236                     if (it != mapZerocoinspends.end()) {
237                         nTimeSeen = it->second;
238                     } else {
239                         //for some reason not in map, add it
240                         mapZerocoinspends[txid] = nTimeSeen;
241                     }
242
243                     double nTimePriority = std::pow(GetAdjustedTime() - nTimeSeen, 6);
244
245                     // zPCH spends can have very large priority, use non-overflowing safe functions
246                     dPriority = double_safe_addition(dPriority, (nTimePriority * nConfs));
247                     dPriority = double_safe_multiplication(dPriority, nTotalIn);
248
249                     continue;
250                 }
251
252                 // Read prev transaction
253                 if (!view.HaveCoins(txin.prevout.hash)) {
254                     // This should never happen; all transactions in the memory
255                     // pool should connect to either transactions in the chain
256                     // or other transactions in the memory pool.
257                     if (!mempool.mapTx.count(txin.prevout.hash)) {
258                         LogPrintf("ERROR: mempool transaction missing input\n");
259                         if (fDebug) assert("mempool transaction missing input" == 0);
260                         fMissingInputs = true;
261                         if (porphan)
262                             vOrphan.pop_back();
263                         break;
264                     }
265
266                     // Has to wait for dependencies
267                     if (!porphan) {
268                         // Use list for automatic deletion
269                         vOrphan.push_back(COrphan(&tx));
270                         porphan = &vOrphan.back();
271                     }
272                     mapDependers[txin.prevout.hash].push_back(porphan);
273                     porphan->setDependsOn.insert(txin.prevout.hash);
274                     nTotalIn += mempool.mapTx[txin.prevout.hash].GetTx().vout[txin.prevout.n].nValue;
275                     continue;
276                 }
277
278                 //Check for invalid/fraudulent inputs. They shouldn't make it through mempool, but check
                        anyways.
279                 if (invalid_out::ContainsOutPoint(txin.prevout)) {
280                     LogPrintf("%s : found invalid input %s in tx %s", __func__, txin.prevout.ToString(),
                            tx.GetHash().ToString());
281                     fMissingInputs = true;
```

```
                    break;
                }

                const CCoins* coins = view.AccessCoins(txin.prevout.hash);
                assert(coins);

                CAmount nValueIn = coins->vout[txin.prevout.n].nValue;
                nTotalIn += nValueIn;

                int nConf = nHeight - coins->nHeight;

                // zPCH spends can have very large priority, use non-overflowing safe functions
                dPriority = double_safe_addition(dPriority, ((double)nValueIn * nConf));

            }
            if (fMissingInputs) continue;

            // Priority is sum(valuein * age) / modified_txsize
            unsigned int nTxSize = ::GetSerializeSize(tx, SER_NETWORK, PROTOCOL_VERSION);
            dPriority = tx.ComputePriority(dPriority, nTxSize);

            uint256 hash = tx.GetHash();
            mempool.ApplyDeltas(hash, dPriority, nTotalIn);

            CFeeRate feeRate(nTotalIn - tx.GetValueOut(), nTxSize);

            if (porphan) {
                porphan->dPriority = dPriority;
                porphan->feeRate = feeRate;
            } else
                vecPriority.push_back(TxPriority(dPriority, feeRate, &mi->second.GetTx()));
        }

        // Collect transactions into block
        uint64_t nBlockSize = 1000;
        uint64_t nBlockTx = 0;
        int nBlockSigOps = 100;
        bool fSortedByFee = (nBlockPrioritySize <= 0);

        TxPriorityCompare comparer(fSortedByFee);
        std::make_heap(vecPriority.begin(), vecPriority.end(), comparer);

        vector<CBigNum> vBlockSerials;
        vector<CBigNum> vTxSerials;
        while (!vecPriority.empty()) {
            // Take highest priority transaction off the priority queue:
            double dPriority = vecPriority.front().get<0>();
            CFeeRate feeRate = vecPriority.front().get<1>();
            const CTransaction& tx = *(vecPriority.front().get<2>());

            std::pop_heap(vecPriority.begin(), vecPriority.end(), comparer);
            vecPriority.pop_back();

            // Size limits
            unsigned int nTxSize = ::GetSerializeSize(tx, SER_NETWORK, PROTOCOL_VERSION);
            if (nBlockSize + nTxSize >= nBlockMaxSize)
                continue;

            // Legacy limits on sigOps:
            unsigned int nMaxBlockSigOps = MAX_BLOCK_SIGOPS_CURRENT;
            unsigned int nTxSigOps = GetLegacySigOpCount(tx);
            if (nBlockSigOps + nTxSigOps >= nMaxBlockSigOps)
```

```
344                 continue;
345
346             // Skip free transactions if we're past the minimum block size:
347             const uint256& hash = tx.GetHash();
348             double dPriorityDelta = 0;
349             CAmount nFeeDelta = 0;
350             mempool.ApplyDeltas(hash, dPriorityDelta, nFeeDelta);
351             if (!tx.HasZerocoinSpendInputs() && fSortedByFee && (dPriorityDelta <= 0) && (nFeeDelta <= 0)
                      && (feeRate < ::minRelayTxFee) && (nBlockSize + nTxSize >= nBlockMinSize))
352                 continue;
353
354             // Prioritise by fee once past the priority size or we run out of high-priority
355             // transactions:
356             if (!fSortedByFee &&
357                 ((nBlockSize + nTxSize >= nBlockPrioritySize) || !AllowFree(dPriority))) {
358                 fSortedByFee = true;
359                 comparer = TxPriorityCompare(fSortedByFee);
360                 std::make_heap(vecPriority.begin(), vecPriority.end(), comparer);
361             }
362
363             if (!view.HaveInputs(tx))
364                 continue;
365
366             // double check that there are no double spent zPCH spends in this block or tx
367             if (tx.HasZerocoinSpendInputs()) {
368                 int nHeightTx = 0;
369                 if (IsTransactionInChain(tx.GetHash(), nHeightTx))
370                     continue;
371
372                 bool fDoubleSerial = false;
373                 for (const CTxIn& txIn : tx.vin) {
374                     bool isPublicSpend = txIn.IsZerocoinPublicSpend();
375                     if (txIn.IsZerocoinSpend() || isPublicSpend) {
376                         libzerocoin::CoinSpend* spend;
377                         if (isPublicSpend) {
378                             libzerocoin::ZerocoinParams* params = Params().Zerocoin_Params(false);
379                             PublicCoinSpend publicSpend(params);
380                             CValidationState state;
381                             if (!ZPCHModule::ParseZerocoinPublicSpend(txIn, tx, state, publicSpend)){
382                                 throw std::runtime_error("Invalid public spend parse");
383                             }
384                             spend = &publicSpend;
385                         } else {
386                             libzerocoin::CoinSpend spendObj = TxInToZerocoinSpend(txIn);
387                             spend = &spendObj;
388                         }
389
390                         bool fUseV1Params = libzerocoin::ExtractVersionFromSerial(spend->
                                 getCoinSerialNumber()) < libzerocoin::PrivateCoin::PUBKEY_VERSION;
391                         if (!spend->HasValidSerial(Params().Zerocoin_Params(fUseV1Params)))
392                             fDoubleSerial = true;
```

3.1.4 CID 242476 - Resource leak

In `src/wallet/rpcwallet.cpp:serchdzpch`, dzpchThreads is dynamically allocated but not freed before control leaves the function. It will cause a memory leak which can disrupt stability.

```
3632 UniValue searchdzpch(const UniValue& params, bool fHelp)
3633 {
```

```
3634     if(fHelp || params.size() != 3)
3635         throw runtime_error(
3636             "searchdzpch\n"
3637             "\nMake an extended search for deterministically generated zPCH that have not yet been
                      recognized by the wallet.\n" +
3638             HelpRequiringPassphrase() + "\n"
3639 
3640             "\nArguments\n"
3641             "1. \"count\"       (numeric) Which sequential zPCH to start with.\n"
3642             "2. \"range\"       (numeric) How many zPCH to generate.\n"
3643             "3. \"threads\"     (numeric) How many threads should this operation consume.\n"
3644 
3645             "\nExamples\n" +
3646             HelpExampleCli("searchdzpch", "1, 100, 2") + HelpExampleRpc("searchdzpch", "1, 100, 2"));
3647 
3648     EnsureWalletIsUnlocked();
3649 
3650     int nCount = params[0].get_int();
3651     if (nCount < 0)
3652         throw JSONRPCError(RPC_INVALID_PARAMETER, "Count cannot be less than 0");
3653 
3654     int nRange = params[1].get_int();
3655     if (nRange < 1)
3656         throw JSONRPCError(RPC_INVALID_PARAMETER, "Range has to be at least 1");
3657 
3658     int nThreads = params[2].get_int();
3659 
3660     CzPCHWallet* zwallet = pwalletMain->zwalletMain;
3661 
3662     boost::thread_group* dzpchThreads = new boost::thread_group();
3663     int nRangePerThread = nRange / nThreads;
3664 
3665     int nPrevThreadEnd = nCount - 1;
3666     for (int i = 0; i < nThreads; i++) {
3667         int nStart = nPrevThreadEnd + 1;;
3668         int nEnd = nStart + nRangePerThread;
3669         nPrevThreadEnd = nEnd;
3670         dzpchThreads->create_thread(boost::bind(&SearchThread, zwallet, nStart, nEnd));
3671     }
3672 
3673     dzpchThreads->join_all();
3674 
3675     zwallet->RemoveMintsFromPool(pwalletMain->zpchTracker->GetSerialHashes());
3676     zwallet->SyncWithChain(false);
3677 
3678     //todo: better response
3679     return "done";
3680 }
```

3.1.5 CID 242474 - Resource leak

In `src/torcontrol.cpp:ReadBinaryFile`, the file is not properly closed if file descriptor encountered an error. It can cause file descriptor leak which can lead to system instability.

```
370 static std::pair<bool,std::string> ReadBinaryFile(const std::string &filename, size_t maxsize=std::
        numeric_limits<size_t>::max())
371 {
372     FILE *f = fopen(filename.c_str(), "rb");
373     if (f == NULL)
```

```
374         return std::make_pair(false,"");
375     std::string retval;
376     char buffer[128];
377     size_t n;
378     while ((n=fread(buffer, 1, sizeof(buffer), f)) > 0) {
379         // Check for reading errors so we don't return any data if we couldn't
380         // read the entire file (or up to maxsize)
381         if (ferror(f))
382             return std::make_pair(false,"");
383         retval.append(buffer, buffer+n);
384         if (retval.size() > maxsize)
385             break;
386     }
387     fclose(f);
388     return std::make_pair(true,retval);
389 }
```

3.1.6 CID 242447 - Uninitialized local variable

In `src/spork.cpp:SetPrivKey` uses an uninitialized local variable to test spork signer. It may lead to an undesirable result. It is strongly advised to use a properly initialized object.

```
255 bool CSporkManager::SetPrivKey(std::string strPrivKey)
256 {
257     CSporkMessage msg;
258 
259     // Test signing successful, proceed
260     strMasterPrivKey = strPrivKey;
261 
262     Sign(msg);
263 
264     if (CheckSignature(msg, true)) {
265         LogPrintf("CSporkManager::SetPrivKey - Successfully initialized as spork signer\n");
266         return true;
267     } else {
268         return false;
269     }
270 }
```

3.1.7 CID 186065 - Out-of-bounds access

In `src/pubkey.cpp:CPubKey::Derive` incorrectly indexes pubkey array which leads to access invalid position(ptr+33 > ptr+32). It may lead into severe data corruption.

```
231 bool CPubKey::Derive(CPubKey& pubkeyChild, ChainCode &ccChild, unsigned int nChild, const ChainCode& cc)
        const
232 {
233     assert(IsValid());
234     assert((nChild >> 31) == 0);
235     assert(size() == COMPRESSED_PUBLIC_KEY_SIZE);
236     unsigned char out[64];
237     BIP32Hash(cc, nChild, *begin(), begin()+1, out);
238     memcpy(ccChild.begin(), out+32, 32);
239     secp256k1_pubkey pubkey;
240     if (!secp256k1_ec_pubkey_parse(secp256k1_context_verify, &pubkey, &(*this)[0], size())) {
241         return false;
242     }
```

```
    if (!secp256k1_ec_pubkey_tweak_add(secp256k1_context_verify, &pubkey, out)) {
        return false;
    }
    unsigned char pub[COMPRESSED_PUBLIC_KEY_SIZE];
    size_t publen = COMPRESSED_PUBLIC_KEY_SIZE;
    secp256k1_ec_pubkey_serialize(secp256k1_context_verify, pub, &publen, &pubkey,
         SECP256K1_EC_COMPRESSED);
    pubkeyChild.Set(pub, pub + publen);
return true;
}
```

3.2 Static analysis using Synopsis® Coverity® Scan - Medium impact issues

3.2.1 CID 242536 - Dereference after null check

In `src/rpc/budget.cpp:preparebudget`, pindexPrev can be null in a special case. Though it will not cause a practical problem in production due to its special condition requirement, it is good to fix this problem to improve code quality.

```
UniValue preparebudget(const UniValue& params, bool fHelp)
{
    int nBlockMin = 0;
    CBlockIndex* pindexPrev = chainActive.Tip();

    if (fHelp || params.size() != 6)
        throw runtime_error(
            "preparebudget \"proposal-name\" \"url\" payment-count block-start \"popchain-address\"
                 monthy-payment\n"
            "\nPrepare proposal for network by signing and creating tx\n"

            "\nArguments:\n"
            "1. \"proposal-name\":  (string, required) Desired proposal name (20 character limit)\n"
            "2. \"url\":            (string, required) URL of proposal details (64 character limit)\n"
            "3. payment-count:    (numeric, required) Total number of monthly payments\n"
            "4. block-start:      (numeric, required) Starting super block height\n"
            "5. \"popchain-address\":   (string, required) PIVX address to send payments to\n"
            "6. monthly-payment:  (numeric, required) Monthly payment amount\n"

            "\nResult:\n"
            "\"xxxx\"       (string) proposal fee hash (if successful) or error message (if failed)\n"

            "\nExamples:\n" +
            HelpExampleCli("preparebudget", "\"test-proposal\" \"https://forum.popchain.org/t/test-
                 proposal\" 2 820800 \"D9oc6C3dttUbv8zd7zGNq1qKBGf4ZQ1XEE\" 500") +
            HelpExampleRpc("preparebudget", "\"test-proposal\" \"https://forum.popchain.org/t/test-
                 proposal\" 2 820800 \"D9oc6C3dttUbv8zd7zGNq1qKBGf4ZQ1XEE\" 500"));

    LOCK2(cs_main, pwalletMain->cs_wallet);

    EnsureWalletIsUnlocked();

    std::string strProposalName = SanitizeString(params[0].get_str());
    if (strProposalName.size() > 20)
        throw runtime_error("Invalid proposal name, limit of 20 characters.");

    std::string strURL = SanitizeString(params[1].get_str());
    if (strURL.size() > 64)
        throw runtime_error("Invalid url, limit of 64 characters.");
```

```
88
89      int nPaymentCount = params[2].get_int();
90      if (nPaymentCount < 1)
91          throw runtime_error("Invalid payment count, must be more than zero.");
92
93      // Start must be in the next budget cycle
94      if (pindexPrev != NULL) nBlockMin = pindexPrev->nHeight - pindexPrev->nHeight % Params().
            GetBudgetCycleBlocks() + Params().GetBudgetCycleBlocks();
95
96      int nBlockStart = params[3].get_int();
97      if (nBlockStart % Params().GetBudgetCycleBlocks() != 0) {
98          int nNext = pindexPrev->nHeight - pindexPrev->nHeight % Params().GetBudgetCycleBlocks() + Params
                ().GetBudgetCycleBlocks();
99          throw runtime_error(strprintf("Invalid block start - must be a budget cycle block. Next valid
                block: %d", nNext));
100     }
101
102     int nBlockEnd = nBlockStart + Params().GetBudgetCycleBlocks() * nPaymentCount; // End must be AFTER
            current cycle
103
104     if (nBlockStart < nBlockMin)
105         throw runtime_error("Invalid block start, must be more than current height.");
106
107     if (nBlockEnd < pindexPrev->nHeight)
108         throw runtime_error("Invalid ending block, starting block + (payment_cycle*payments) must be more
                than current height.");
109
110     CBitcoinAddress address(params[4].get_str());
```

3.2.2 CID 242518 - Dereference after null check

In `src/zpch/zpchwallet.cpp:CzPCHWallet::SyncWithChain`, in case of pindex is null (some special condition is required), it can cause wallet crash.

```
182 void CzPCHWallet::SyncWithChain(bool fGenerateMintPool)
183 {
184     uint32_t nLastCountUsed = 0;
185     bool found = true;
186     CWalletDB walletdb(strWalletFile);
187
188     set<uint256> setAddedTx;
189     while (found) {
190         found = false;
191         if (fGenerateMintPool)
192             GenerateMintPool();
193         LogPrintf("%s: Mintpool size=%d\n", __func__, mintPool.size());
194
195         std::set<uint256> setChecked;
196         list<pair<uint256,uint32_t> > listMints = mintPool.List();
197         for (pair<uint256, uint32_t> pMint : listMints) {
198             LOCK(cs_main);
199             if (setChecked.count(pMint.first))
200                 return;
201             setChecked.insert(pMint.first);
202
203             if (ShutdownRequested())
204                 return;
205
206             if (pwalletMain->zpchTracker->HasPubcoinHash(pMint.first)) {
207                 mintPool.Remove(pMint.first);
```

```
                continue;
            }

            uint256 txHash;
            CZerocoinMint mint;
            if (zerocoinDB->ReadCoinMint(pMint.first, txHash)) {
                //this mint has already occurred on the chain, increment counter's state to reflect this
                LogPrintf("%s : Found wallet coin mint=%s count=%d tx=%s\n", __func__, pMint.first.GetHex
                     (), pMint.second, txHash.GetHex());
                found = true;

                uint256 hashBlock;
                CTransaction tx;
                if (!GetTransaction(txHash, tx, hashBlock, true)) {
                    LogPrintf("%s : failed to get transaction for mint %s!\n", __func__, pMint.first.
                         GetHex());
                    found = false;
                    nLastCountUsed = std::max(pMint.second, nLastCountUsed);
                    continue;
                }

                //Find the denomination
                CoinDenomination denomination = CoinDenomination::ZQ_ERROR;
                bool fFoundMint = false;
                CBigNum bnValue = 0;
                for (const CTxOut& out : tx.vout) {
                    if (!out.IsZerocoinMint())
                        continue;

                    PublicCoin pubcoin(Params().Zerocoin_Params(false));
                    CValidationState state;
                    if (!TxOutToPublicCoin(out, pubcoin, state)) {
                        LogPrintf("%s : failed to get mint from txout for %s!\n", __func__, pMint.first.
                             GetHex());
                        continue;
                    }

                    // See if this is the mint that we are looking for
                    uint256 hashPubcoin = GetPubCoinHash(pubcoin.getValue());
                    if (pMint.first == hashPubcoin) {
                        denomination = pubcoin.getDenomination();
                        bnValue = pubcoin.getValue();
                        fFoundMint = true;
                        break;
                    }
                }

                if (!fFoundMint || denomination == ZQ_ERROR) {
                    LogPrintf("%s : failed to get mint %s from tx %s!\n", __func__, pMint.first.GetHex(),
                          tx.GetHash().GetHex());
                    found = false;
                    break;
                }

                CBlockIndex* pindex = nullptr;
                if (mapBlockIndex.count(hashBlock))
                    pindex = mapBlockIndex.at(hashBlock);

                if (!setAddedTx.count(txHash)) {
                    CBlock block;
                    CWalletTx wtx(pwalletMain, tx);
                    if (pindex && ReadBlockFromDisk(block, pindex))
```

```
                        wtx.SetMerkleBranch(block);

                    //Fill out wtx so that a transaction record can be created
                    wtx.nTimeReceived = pindex->GetBlockTime();
                    pwalletMain->AddToWallet(wtx);
                    setAddedTx.insert(txHash);
                }

                SetMintSeen(bnValue, pindex->nHeight, txHash, denomination);
                nLastCountUsed = std::max(pMint.second, nLastCountUsed);
                nCountLastUsed = std::max(nLastCountUsed, nCountLastUsed);
                LogPrint("zero", "%s: updated count to %d\n", __func__, nCountLastUsed);
            }
        }
    }
}
```

3.2.3 CID 242507 - Explicit null dereferenced

In `src/zpch/zpchwallet.cpp:CzPCHWallet::SyncWithChain`, in special case if `mapBlockIndex.count(hashBlock)` is zero, `pindex` will be null. In this case, it will lead to wallet crash.

```
void CzPCHWallet::SyncWithChain(bool fGenerateMintPool)
{
    uint32_t nLastCountUsed = 0;
    bool found = true;
    CWalletDB walletdb(strWalletFile);

    set<uint256> setAddedTx;
    while (found) {
        found = false;
        if (fGenerateMintPool)
            GenerateMintPool();
        LogPrintf("%s: Mintpool size=%d\n", __func__, mintPool.size());

        std::set<uint256> setChecked;
        list<pair<uint256,uint32_t> > listMints = mintPool.List();
        for (pair<uint256, uint32_t> pMint : listMints) {
            LOCK(cs_main);
            if (setChecked.count(pMint.first))
                return;
            setChecked.insert(pMint.first);

            if (ShutdownRequested())
                return;

            if (pwalletMain->zpchTracker->HasPubcoinHash(pMint.first)) {
                mintPool.Remove(pMint.first);
                continue;
            }

            uint256 txHash;
            CZerocoinMint mint;
            if (zerocoinDB->ReadCoinMint(pMint.first, txHash)) {
                //this mint has already occurred on the chain, increment counter's state to reflect this
                LogPrintf("%s : Found wallet coin mint=%s count=%d tx=%s\n", __func__, pMint.first.GetHex
                    (), pMint.second, txHash.GetHex());
                found = true;

                uint256 hashBlock;
```

```
                CTransaction tx;
                if (!GetTransaction(txHash, tx, hashBlock, true)) {
                    LogPrintf("%s : failed to get transaction for mint %s!\n", __func__, pMint.first.
                         GetHex());
                    found = false;
                    nLastCountUsed = std::max(pMint.second, nLastCountUsed);
                    continue;
                }

                //Find the denomination
                CoinDenomination denomination = CoinDenomination::ZQ_ERROR;
                bool fFoundMint = false;
                CBigNum bnValue = 0;
                for (const CTxOut& out : tx.vout) {
                    if (!out.IsZerocoinMint())
                        continue;

                    PublicCoin pubcoin(Params().Zerocoin_Params(false));
                    CValidationState state;
                    if (!TxOutToPublicCoin(out, pubcoin, state)) {
                        LogPrintf("%s : failed to get mint from txout for %s!\n", __func__, pMint.first.
                             GetHex());
                        continue;
                    }

                    // See if this is the mint that we are looking for
                    uint256 hashPubcoin = GetPubCoinHash(pubcoin.getValue());
                    if (pMint.first == hashPubcoin) {
                        denomination = pubcoin.getDenomination();
                        bnValue = pubcoin.getValue();
                        fFoundMint = true;
                        break;
                    }
                }

                if (!fFoundMint || denomination == ZQ_ERROR) {
                    LogPrintf("%s : failed to get mint %s from tx %s!\n", __func__, pMint.first.GetHex(),
                          tx.GetHash().GetHex());
                    found = false;
                    break;
                }

                CBlockIndex* pindex = nullptr;
                if (mapBlockIndex.count(hashBlock))
                    pindex = mapBlockIndex.at(hashBlock);

                if (!setAddedTx.count(txHash)) {
                    CBlock block;
                    CWalletTx wtx(pwalletMain, tx);
                    if (pindex && ReadBlockFromDisk(block, pindex))
                        wtx.SetMerkleBranch(block);

                    //Fill out wtx so that a transaction record can be created
                    wtx.nTimeReceived = pindex->GetBlockTime();
                    pwalletMain->AddToWallet(wtx);
                    setAddedTx.insert(txHash);
                }

                SetMintSeen(bnValue, pindex->nHeight, txHash, denomination);
                nLastCountUsed = std::max(pMint.second, nLastCountUsed);
                nCountLastUsed = std::max(nLastCountUsed, nCountLastUsed);
                LogPrint("zero", "%s: updated count to %d\n", __func__, nCountLastUsed);
```

```
278             }
279         }
280     }
281 }
```

3.2.4 CID 242505 - Explicit null dereferenced

In `src/main.cpp:AcceptBlock`, in special case when handling genesis block and zerocoin is active, it will lead into null pointer dereference. In this case, it will lead to a wallet crash.

```
182 void CzPCHWallet::SyncWithChain(bool fGenerateMintPool)
183 {
184     uint32_t nLastCountUsed = 0;
185     bool found = true;
186     CWalletDB walletdb(strWalletFile);
187 
188     set<uint256> setAddedTx;
189     while (found) {
190         found = false;
191         if (fGenerateMintPool)
192             GenerateMintPool();
193         LogPrintf("%s: Mintpool size=%d\n", __func__, mintPool.size());
194 
195         std::set<uint256> setChecked;
196         list<pair<uint256,uint32_t> > listMints = mintPool.List();
197         for (pair<uint256, uint32_t> pMint : listMints) {
198             LOCK(cs_main);
199             if (setChecked.count(pMint.first))
200                 return;
201             setChecked.insert(pMint.first);
202 
203             if (ShutdownRequested())
204                 return;
205 
206             if (pwalletMain->zpchTracker->HasPubcoinHash(pMint.first)) {
207                 mintPool.Remove(pMint.first);
208                 continue;
209             }
210 
211             uint256 txHash;
212             CZerocoinMint mint;
213             if (zerocoinDB->ReadCoinMint(pMint.first, txHash)) {
214                 //this mint has already occurred on the chain, increment counter's state to reflect this
215                 LogPrintf("%s : Found wallet coin mint=%s count=%d tx=%s\n", __func__, pMint.first.GetHex
                        (), pMint.second, txHash.GetHex());
216                 found = true;
217 
218                 uint256 hashBlock;
219                 CTransaction tx;
220                 if (!GetTransaction(txHash, tx, hashBlock, true)) {
221                     LogPrintf("%s : failed to get transaction for mint %s!\n", __func__, pMint.first.
                            GetHex());
222                     found = false;
223                     nLastCountUsed = std::max(pMint.second, nLastCountUsed);
224                     continue;
225                 }
226 
227                 //Find the denomination
228                 CoinDenomination denomination = CoinDenomination::ZQ_ERROR;
```

```
229                 bool fFoundMint = false;
230                 CBigNum bnValue = 0;
231                 for (const CTxOut& out : tx.vout) {
232                     if (!out.IsZerocoinMint())
233                         continue;
234 
235                     PublicCoin pubcoin(Params().Zerocoin_Params(false));
236                     CValidationState state;
237                     if (!TxOutToPublicCoin(out, pubcoin, state)) {
238                         LogPrintf("%s : failed to get mint from txout for %s!\n", __func__, pMint.first.
                                GetHex());
239                         continue;
240                     }
241 
242                     // See if this is the mint that we are looking for
243                     uint256 hashPubcoin = GetPubCoinHash(pubcoin.getValue());
244                     if (pMint.first == hashPubcoin) {
245                         denomination = pubcoin.getDenomination();
246                         bnValue = pubcoin.getValue();
247                         fFoundMint = true;
248                         break;
249                     }
250                 }
251 
252                 if (!fFoundMint || denomination == ZQ_ERROR) {
253                     LogPrintf("%s : failed to get mint %s from tx %s!\n", __func__, pMint.first.GetHex(),
                            tx.GetHash().GetHex());
254                     found = false;
255                     break;
256                 }
257 
258                 CBlockIndex* pindex = nullptr;
259                 if (mapBlockIndex.count(hashBlock))
260                     pindex = mapBlockIndex.at(hashBlock);
261 
262                 if (!setAddedTx.count(txHash)) {
263                     CBlock block;
264                     CWalletTx wtx(pwalletMain, tx);
265                     if (pindex && ReadBlockFromDisk(block, pindex))
266                         wtx.SetMerkleBranch(block);
267 
268                     //Fill out wtx so that a transaction record can be created
269                     wtx.nTimeReceived = pindex->GetBlockTime();
270                     pwalletMain->AddToWallet(wtx);
271                     setAddedTx.insert(txHash);
272                 }
273 
274                 SetMintSeen(bnValue, pindex->nHeight, txHash, denomination);
275                 nLastCountUsed = std::max(pMint.second, nLastCountUsed);
276                 nCountLastUsed = std::max(nLastCountUsed, nCountLastUsed);
277                 LogPrint("zero", "%s: updated count to %d\n", __func__, nCountLastUsed);
278             }
279         }
280     }
281 }
```

3.2.5 CID 242494 - Dereference after null check

In `src/rpc/budget.cpp:submitbudget`, in speical case when handling genesis block, it can lead into null pointer dereference. In this case, it will lead to a wallet crash.

```
147 UniValue submitbudget(const UniValue& params, bool fHelp)
148 {
149     int nBlockMin = 0;
150     CBlockIndex* pindexPrev = chainActive.Tip();
151 
152     if (fHelp || params.size() != 7)
153         throw runtime_error(
154             "submitbudget \"proposal-name\" \"url\" payment-count block-start \"popchain-address\" monthy
                -payment \"fee-tx\"\n"
155             "\nSubmit proposal to the network\n"
156 
157             "\nArguments:\n"
158             "1. \"proposal-name\":  (string, required) Desired proposal name (20 character limit)\n"
159             "2. \"url\":            (string, required) URL of proposal details (64 character limit)\n"
160             "3. payment-count:    (numeric, required) Total number of monthly payments\n"
161             "4. block-start:      (numeric, required) Starting super block height\n"
162             "5. \"popchain-address\":   (string, required) PIVX address to send payments to\n"
163             "6. monthly-payment:  (numeric, required) Monthly payment amount\n"
164             "7. \"fee-tx\":         (string, required) Transaction hash from preparebudget command\n"
165 
166             "\nResult:\n"
167             "\"xxxx\"       (string) proposal hash (if successful) or error message (if failed)\n"
168 
169             "\nExamples:\n" +
170             HelpExampleCli("submitbudget", "\"test-proposal\" \"https://forum.popchain.org/t/test-
                proposal\" 2 820800 \"D9oc6C3dttUbv8zd7zGNq1qKBGf4ZQ1XEE\" 500") +
171             HelpExampleRpc("submitbudget", "\"test-proposal\" \"https://forum.popchain.org/t/test-
                proposal\" 2 820800 \"D9oc6C3dttUbv8zd7zGNq1qKBGf4ZQ1XEE\" 500"));
172 
173     // Check these inputs the same way we check the vote commands:
174     // **********************************************************
175 
176     std::string strProposalName = SanitizeString(params[0].get_str());
177     if (strProposalName.size() > 20)
178         throw runtime_error("Invalid proposal name, limit of 20 characters.");
179 
180     std::string strURL = SanitizeString(params[1].get_str());
181     if (strURL.size() > 64)
182         throw runtime_error("Invalid url, limit of 64 characters.");
183 
184     int nPaymentCount = params[2].get_int();
185     if (nPaymentCount < 1)
186         throw runtime_error("Invalid payment count, must be more than zero.");
187 
188     // Start must be in the next budget cycle
189     if (pindexPrev != NULL) nBlockMin = pindexPrev->nHeight - pindexPrev->nHeight % Params().
          GetBudgetCycleBlocks() + Params().GetBudgetCycleBlocks();
190 
191     int nBlockStart = params[3].get_int();
192     if (nBlockStart % Params().GetBudgetCycleBlocks() != 0) {
193         int nNext = pindexPrev->nHeight - pindexPrev->nHeight % Params().GetBudgetCycleBlocks() + Params
              ().GetBudgetCycleBlocks();
194         throw runtime_error(strprintf("Invalid block start - must be a budget cycle block. Next valid
              block: %d", nNext));
195     }
196 
197     int nBlockEnd = nBlockStart + (Params().GetBudgetCycleBlocks() * nPaymentCount); // End must be AFTER
          current cycle
198 
199     if (nBlockStart < nBlockMin)
200         throw runtime_error("Invalid block start, must be more than current height.");
201 
```

```
202     if (nBlockEnd < pindexPrev->nHeight)
203         throw runtime_error("Invalid ending block, starting block + (payment_cycle*payments) must be more
                than current height.");
204
205     CBitcoinAddress address(params[4].get_str());
206     if (!address.IsValid())
207         throw JSONRPCError(RPC_INVALID_ADDRESS_OR_KEY, "Invalid PIVX address");
208
209     // Parse PIVX address
210     CScript scriptPubKey = GetScriptForDestination(address.Get());
211     CAmount nAmount = AmountFromValue(params[5]);
212     uint256 hash = ParseHashV(params[6], "parameter 1");
213
214     //create the proposal incase we're the first to make it
215     CBudgetProposalBroadcast budgetProposalBroadcast(strProposalName, strURL, nPaymentCount, scriptPubKey
            , nAmount, nBlockStart, hash);
216
217     std::string strError = "";
218     int nConf = 0;
219     if (!IsBudgetCollateralValid(hash, budgetProposalBroadcast.GetHash(), strError,
            budgetProposalBroadcast.nTime, nConf)) {
220         throw runtime_error("Proposal FeeTX is not valid - " + hash.ToString() + " - " + strError);
221     }
222
223     if (!masternodeSync.IsBlockchainSynced()) {
224         throw runtime_error("Must wait for client to sync with masternode network. Try again in a minute
                or so.");
225     }
226
227     // if(!budgetProposalBroadcast.IsValid(strError)){
228     //     return "Proposal is not valid - " + budgetProposalBroadcast.GetHash().ToString() + " - " +
            strError;
229     // }
230
231     budget.mapSeenMasternodeBudgetProposals.insert(make_pair(budgetProposalBroadcast.GetHash(),
            budgetProposalBroadcast));
232     budgetProposalBroadcast.Relay();
233     if(budget.AddProposal(budgetProposalBroadcast)) {
234         return budgetProposalBroadcast.GetHash().ToString();
235     }
236     throw runtime_error("Invalid proposal, see debug.log for details.");
237 }
```

3.2.6 CID 242485 - Dereference null return value

In `src/rpc/rawtransaction.cpp:signrawtransaction`, when wallet is in regression test mode, there is possibility to crash wallet with crafted request. This vulnerability will not affect production environment because regression test cannot be connected to internet but it is good to fix this problem to improve code quality.

```
630
631 UniValue getbudgetinfo(const UniValue& params, bool fHelp)
632 {
633     if (fHelp || params.size() > 1)
634         throw runtime_error(
635             "getbudgetinfo ( \"proposal\" )\n"
636             "\nShow current masternode budgets\n"
637
638             "\nArguments:\n"
639             "1. \"proposal\"    (string, optional) Proposal name\n"
```

```
            "\nResult:\n"
            "[\n"
            "  {\n"
            "    \"Name\": \"xxxx\",              (string) Proposal Name\n"
            "    \"URL\": \"xxxx\",               (string) Proposal URL\n"
            "    \"Hash\": \"xxxx\",              (string) Proposal vote hash\n"
            "    \"FeeHash\": \"xxxx\",           (string) Proposal fee hash\n"
            "    \"BlockStart\": n,             (numeric) Proposal starting block\n"
            "    \"BlockEnd\": n,               (numeric) Proposal ending block\n"
            "    \"TotalPaymentCount\": n,      (numeric) Number of payments\n"
            "    \"RemainingPaymentCount\": n,  (numeric) Number of remaining payments\n"
            "    \"PaymentAddress\": \"xxxx\",    (string) PIVX address of payment\n"
            "    \"Ratio\": x.xxx,              (numeric) Ratio of yeas vs nays\n"
            "    \"Yeas\": n,                   (numeric) Number of yea votes\n"
            "    \"Nays\": n,                   (numeric) Number of nay votes\n"
            "    \"Abstains\": n,               (numeric) Number of abstains\n"
            "    \"TotalPayment\": xxx.xxx,     (numeric) Total payment amount\n"
            "    \"MonthlyPayment\": xxx.xxx,   (numeric) Monthly payment amount\n"
            "    \"IsEstablished\": true|false, (boolean) Established (true) or (false)\n"
            "    \"IsValid\": true|false,       (boolean) Valid (true) or Invalid (false)\n"
            "    \"IsValidReason\": \"xxxx\",     (string) Error message, if any\n"
            "    \"fValid\": true|false,        (boolean) Valid (true) or Invalid (false)\n"
            "  }\n"
            "  ,...\n"
            "]\n"

            "\nExamples:\n" +
            HelpExampleCli("getbudgetprojection", "") + HelpExampleRpc("getbudgetprojection", ""));

    UniValue ret(UniValue::VARR);

    std::string strShow = "valid";
    if (params.size() == 1) {
        std::string strProposalName = SanitizeString(params[0].get_str());
        CBudgetProposal* pbudgetProposal = budget.FindProposal(strProposalName);
        if (pbudgetProposal == NULL) throw runtime_error("Unknown proposal name");
        UniValue bObj(UniValue::VOBJ);
        budgetToJSON(pbudgetProposal, bObj);
        ret.push_back(bObj);
        return ret;
    }

    std::vector<CBudgetProposal*> winningProps = budget.GetAllProposals();
    for (CBudgetProposal* pbudgetProposal : winningProps) {
        if (strShow == "valid" && !pbudgetProposal->fValid) continue;

        UniValue bObj(UniValue::VOBJ);
        budgetToJSON(pbudgetProposal, bObj);

        ret.push_back(bObj);
    }

    return ret;
}

UniValue mnbudgetrawvote(const UniValue& params, bool fHelp)
{
    if (fHelp || params.size() != 6)
        throw runtime_error(
            "mnbudgetrawvote \"masternode-tx-hash\" masternode-tx-index \"proposal-hash\" yes|no time \"
                 vote-sig\"\n"
```

```
            "\nCompile and relay a proposal vote with provided external signature instead of signing vote
                  internally\n"

            "\nArguments:\n"
            "1. \"masternode-tx-hash\"  (string, required) Transaction hash for the masternode\n"
            "2. masternode-tx-index   (numeric, required) Output index for the masternode\n"
            "3. \"proposal-hash\"       (string, required) Proposal vote hash\n"
            "4. yes|no                (boolean, required) Vote to cast\n"
            "5. time                  (numeric, required) Time since epoch in seconds\n"
            "6. \"vote-sig\"            (string, required) External signature\n"

            "\nResult:\n"
            "\"status\"     (string) Vote status or error message\n"

            "\nExamples:\n" +
            HelpExampleCli("mnbudgetrawvote", "") + HelpExampleRpc("mnbudgetrawvote", ""));

    uint256 hashMnTx = ParseHashV(params[0], "mn tx hash");
    int nMnTxIndex = params[1].get_int();
    CTxIn vin = CTxIn(hashMnTx, nMnTxIndex);

    uint256 hashProposal = ParseHashV(params[2], "Proposal hash");
    std::string strVote = params[3].get_str();

    if (strVote != "yes" && strVote != "no") return "You can only vote 'yes' or 'no'";
    int nVote = VOTE_ABSTAIN;
    if (strVote == "yes") nVote = VOTE_YES;
    if (strVote == "no") nVote = VOTE_NO;

    int64_t nTime = params[4].get_int64();
    std::string strSig = params[5].get_str();
    bool fInvalid = false;
    vector<unsigned char> vchSig = DecodeBase64(strSig.c_str(), &fInvalid);

    if (fInvalid)
        throw JSONRPCError(RPC_INVALID_ADDRESS_OR_KEY, "Malformed base64 encoding");

    CMasternode* pmn = mnodeman.Find(vin);
    if (pmn == NULL) {
        return "Failure to find masternode in list : " + vin.ToString();
    }

    CBudgetVote vote(vin, hashProposal, nVote);
    vote.nTime = nTime;
    vote.vchSig = vchSig;

    if (!vote.SignatureValid(true)) {
        return "Failure to verify signature.";
    }

    std::string strError = "";
    if (budget.UpdateProposal(vote, NULL, strError)) {
        budget.mapSeenMasternodeBudgetVotes.insert(make_pair(vote.GetHash(), vote));
        vote.Relay();
        return "Voted successfully";
    } else {
        return "Error voting : " + strError;
    }
}

UniValue mnfinalbudget(const UniValue& params, bool fHelp)
{
```

```
762     string strCommand;
763     if (params.size() >= 1)
764         strCommand = params[0].get_str();
765 
766     if (fHelp ||
767         (strCommand != "suggest" && strCommand != "vote-many" && strCommand != "vote" && strCommand != "
              show" && strCommand != "getvotes"))
768         throw runtime_error(
769             "mnfinalbudget \"command\"... ( \"passphrase\" )\n"
770             "\nVote or show current budgets\n"
771 
772             "\nAvailable commands:\n"
773             "  vote-many   - Vote on a finalized budget\n"
774             "  vote        - Vote on a finalized budget\n"
775             "  show        - Show existing finalized budgets\n"
776             "  getvotes     - Get vote information for each finalized budget\n");
777 
778     if (strCommand == "vote-many") {
779         if (params.size() != 2)
780             throw runtime_error("Correct usage is 'mnfinalbudget vote-many BUDGET_HASH'");
```

3.2.7 CID 242477 - Dereference after null check

In `src/main.cpp:ConnectBlock`, in practically impossible case when wallet tries to connect genesis block to somewhere, it can lead into null pointer dereference. It is not possible to trigger crash using this problem but it is good to fix this problem to improve code quality.

```
3287 
3288             // Check that zPCH mints are not already known
3289             if (tx.HasZerocoinMintOutputs()) {
3290                 for (auto& out : tx.vout) {
3291                     if (!out.IsZerocoinMint())
3292                         continue;
3293 
3294                     PublicCoin coin(Params().Zerocoin_Params(false));
3295                     if (!TxOutToPublicCoin(out, coin, state))
3296                         return state.DoS(100, error("%s: failed final check of zerocoinmint for tx %s",
                              __func__, tx.GetHash().GetHex()));
3297 
3298                     if (!ContextualCheckZerocoinMint(tx, coin, pindex))
3299                         return state.DoS(100, error("%s: zerocoin mint failed contextual check", __func__
                              ));
3300 
3301                     vMints.emplace_back(make_pair(coin, tx.GetHash()));
3302                 }
3303             }
3304         } else if (!tx.IsCoinBase()) {
3305             if (!view.HaveInputs(tx))
3306                 return state.DoS(100, error("ConnectBlock() : inputs missing/spent"),
3307                     REJECT_INVALID, "bad-txns-inputs-missingorspent");
3308 
3309             // Check that the inputs are not marked as invalid/fraudulent
3310             for (CTxIn in : tx.vin) {
3311                 if (!ValidOutPoint(in.prevout, pindex->nHeight)) {
3312                     return state.DoS(100, error("%s : tried to spend invalid input %s in tx %s", __func__
                          , in.prevout.ToString(),
3313                             tx.GetHash().GetHex()), REJECT_INVALID, "bad-txns-invalid-inputs");
3314                 }
3315             }
```

```
3316
3317            // Check that zPCH mints are not already known
3318            if (tx.HasZerocoinMintOutputs()) {
3319                for (auto& out : tx.vout) {
3320                    if (!out.IsZerocoinMint())
3321                        continue;
3322
3323                    PublicCoin coin(Params().Zerocoin_Params(false));
3324                    if (!TxOutToPublicCoin(out, coin, state))
3325                        return state.DoS(100, error("%s: failed final check of zerocoinmint for tx %s",
                                __func__, tx.GetHash().GetHex()));
3326
3327                    if (!ContextualCheckZerocoinMint(tx, coin, pindex))
3328                        return state.DoS(100, error("%s: zerocoin mint failed contextual check", __func__
                                ));
3329
3330                    vMints.emplace_back(make_pair(coin, tx.GetHash()));
3331                }
3332            }
```

3.2.8 CID 16467 - Improper use of negative value

In `src/allocators.cpp:GetSystemPageSize`, if there is no PAGESIZE defined in `limits.h`, it falls back to `sysconf(3)` syscall. In case of `sysconf(3)` fails, it returns negative number which will cause severe misbehavior of wallet.

```
31  static inline size_t GetSystemPageSize()
32  {
33      size_t page_size;
34  #if defined(WIN32)
35      SYSTEM_INFO sSysInfo;
36      GetSystemInfo(&sSysInfo);
37      page_size = sSysInfo.dwPageSize;
38  #elif defined(PAGESIZE) // defined in limits.h
39      page_size = PAGESIZE;
40  #else                   // assume some POSIX OS
41      page_size = sysconf(_SC_PAGESIZE);
42  #endif
43      return page_size;
44  }
```