# POPCHAIN PROJECT AUDIT REPORT

Version 1.0.0

--

Presented by

**Yong-hyu Ban**
Developer, SCV SOFT

**Benjamin Hyokeun Oh**
CEO, SCV SOFT

Aug 21st, 2019

# Contents

## Introduction

This document includes the results of the audit performed by the SCV SOFT at the request of the POPCHAIN team. And those issues we found has been fixed by the POPCHAIN development ream. The audited report can be found in the public pop-core Github repository, and the version used for this report is commit.

## Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

# Executive Summary

The goal of this audit is to review POPCHAIN's solidity implementation for its decentralized prediction market, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

During the assessment, SCV SOFT identified 7 high-impacted issues and 8 medium-impacted issues. The issues identified during the assessment do not lead to the compilation of vulnerable code. Those issues had been fixed and applied to the POPCHAIN mainnet.

# Summary of Analysis

In this report, we performed our audit according to the procedure described below. Our audit is targeted the full POPCHAIN project except for BerkeleyDB and libboost component.

**The audit showed POPCHAIN mainnet has no critical issues.** And also, those issues have been fixed by POPCHAIN team and SCV SOFT.

| ID | Description | Status |
|---|---|---|
| CID 242525 | Pointer to local outside scope (use-after-free) | Fixed |
| CID242511 | Resource leak | Fixed |
| CID 242479 | Pointer to local outside scope (use-after-free) | Fixed |
| CID242476 | Resource leak | Fixed |
| CID242474 | Resource leak | Fixed |
| CID242447 | Uninitialized local variable | Fixed |
| CID186065 | Out-of-bounds access | Fixed |
| CID242536 | Dereference after null check | Fixed |
| CID242518 | Dereference after null check | Fixed |
| CID242507 | Explicit null dereferenced | Fixed |
| CID242505 | Explicit null dereferenced | Fixed |
| CID242494 | Dereference after null check | Fixed |
| CID242485 | Dereference null return value | Fixed |
| CID242477 | Dereference after null check | Fixed |
| CID16467 | Improper use of negative value | Fixed |

# Analysis Methods

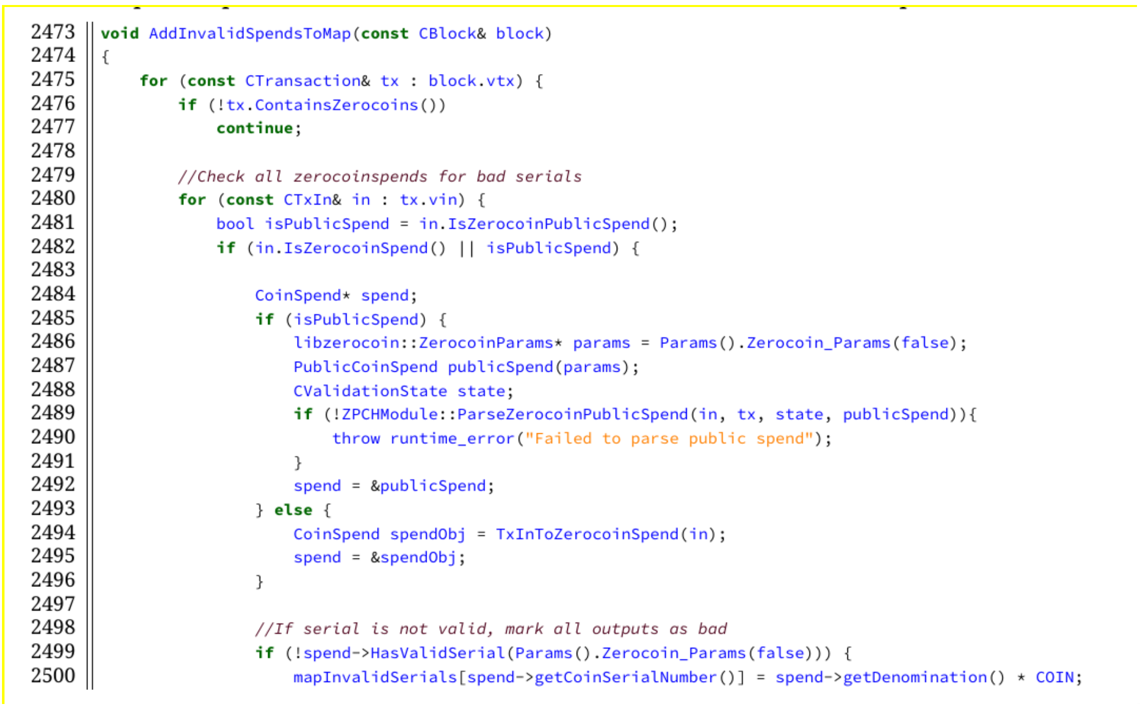
- Full project static analysis using Synopsis® Coverity® Scan
- Manual code inspection

# Analysis Result

## 1. Static analysis using Synopsis® Coverity® Scan - High impact issues

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

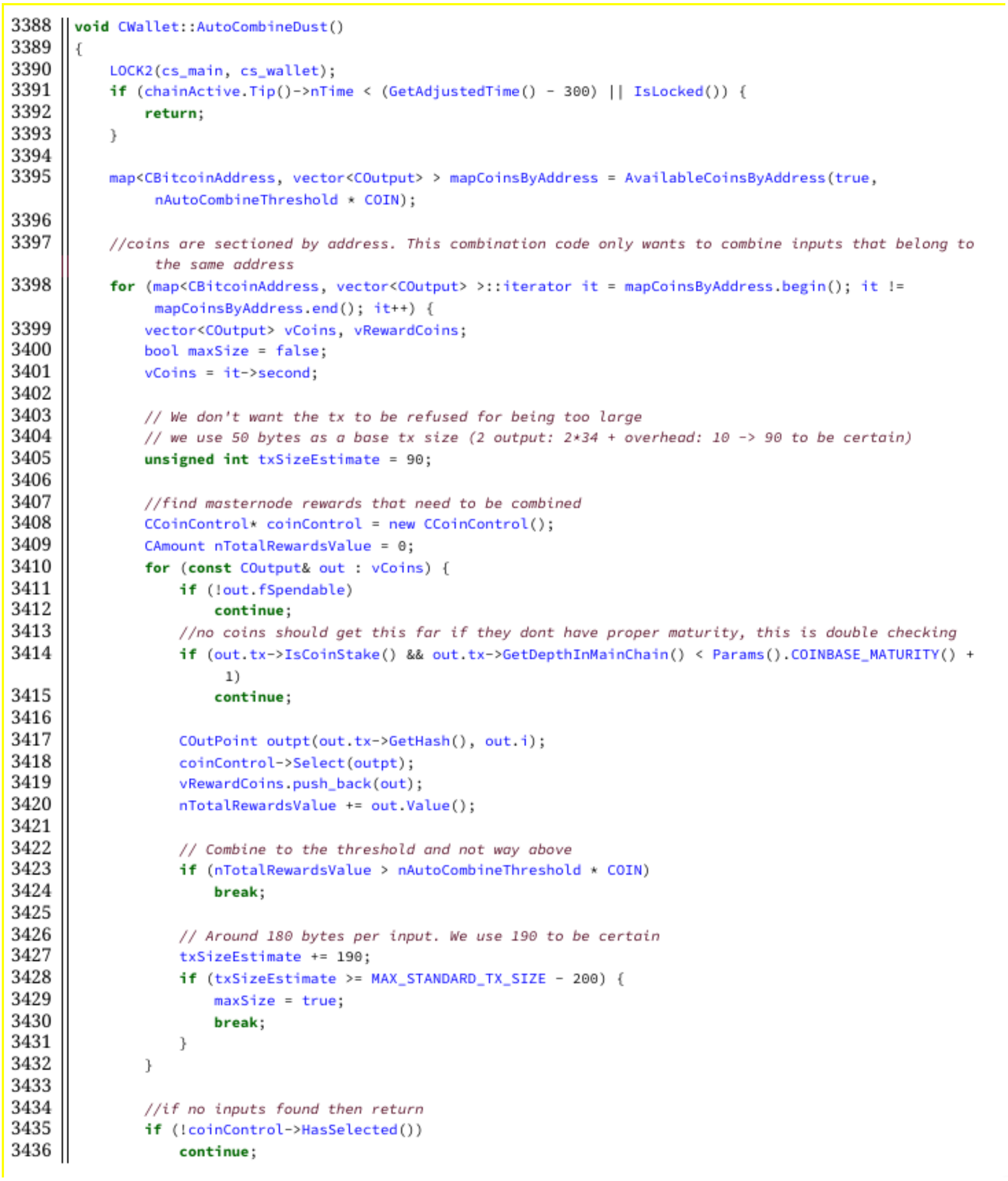
In **src/main.cpp:AddInvalidSpendsToMap**, reference to 'publicSpend' local variable is used after 'publicSpend' is freed from the stack. It can be led to data corruption.



```
void AddInvalidSpendsToMap(const CBlock& block)
{
    for (const CTransaction& tx : block.vtx) {
        if (!tx.ContainsZerocoins())
            continue;

        //Check all zerocoinspends for bad serials
        for (const CTxIn& in : tx.vin) {
            bool isPublicSpend = in.IsZerocoinPublicSpend();
            if (in.IsZerocoinSpend() || isPublicSpend) {

                CoinSpend* spend;
                if (isPublicSpend) {
                    libzerocoin::ZerocoinParams* params = Params().Zerocoin_Params(false);
                    PublicCoinSpend publicSpend(params);
                    CValidationState state;
                    if (!ZPCHModule::ParseZerocoinPublicSpend(in, tx, state, publicSpend)){
                        throw runtime_error("Failed to parse public spend");
                    }
                    spend = &publicSpend;
                } else {
                    CoinSpend spendObj = TxInToZerocoinSpend(in);
                    spend = &spendObj;
                }

                //If serial is not valid, mark all outputs as bad
                if (!spend->HasValidSerial(Params().Zerocoin_Params(false))) {
                    mapInvalidSerials[spend->getCoinSerialNumber()] = spend->getDenomination() * COIN;
```

## 1.2 CID 242511 - Resource leak

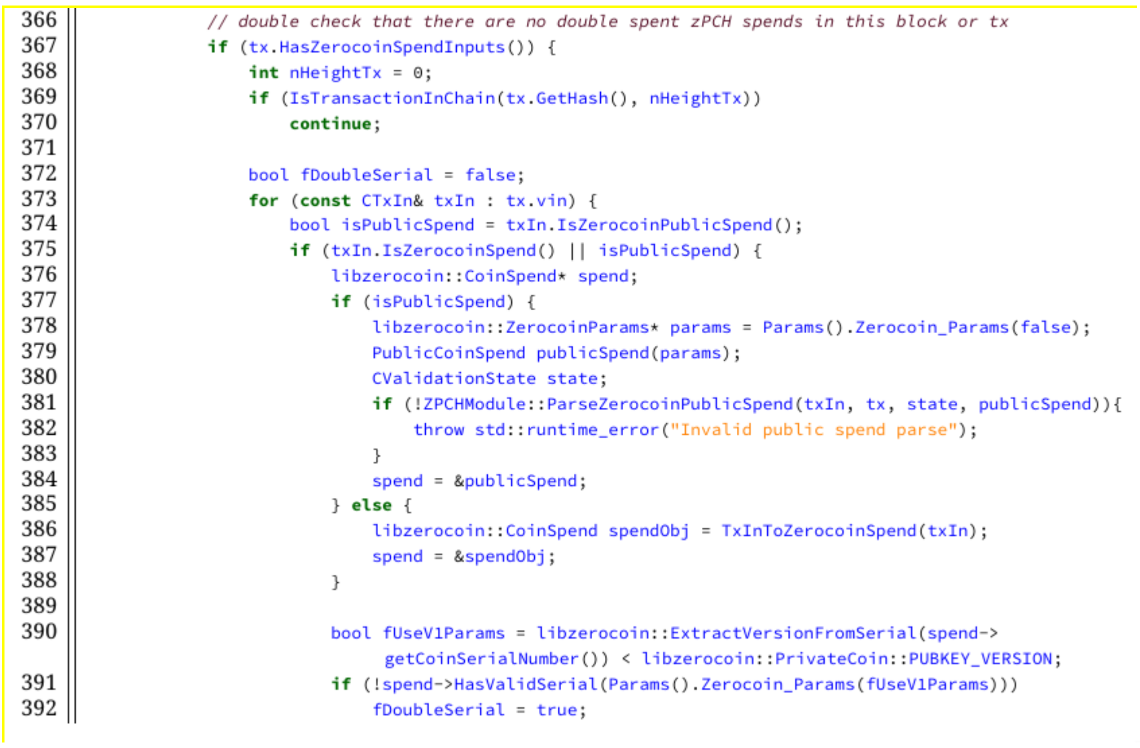
In **src/wallet/wallet.cpp:CWallet::AutoCombineDust**, 'coinControl' is dynamically allocated but not freed when coinControl->HasSelected() is false. It may lead to a memory leak which can disrupt stability.



```
void CWallet::AutoCombineDust()
{
    LOCK2(cs_main, cs_wallet);
    if (chainActive.Tip()->nTime < (GetAdjustedTime() - 300) || IsLocked()) {
        return;
    }

    map<CBitcoinAddress, vector<COutput> > mapCoinsByAddress = AvailableCoinsByAddress(true, nAutoCombineThreshold * COIN);

    //coins are sectioned by address. This combination code only wants to combine inputs that belong to the same address
    for (map<CBitcoinAddress, vector<COutput> >::iterator it = mapCoinsByAddress.begin(); it != mapCoinsByAddress.end(); it++) {
        vector<COutput> vCoins, vRewardCoins;
        bool maxSize = false;
        vCoins = it->second;

        // We don't want the tx to be refused for being too large
        // we use 50 bytes as a base tx size (2 output: 2*34 + overhead: 10 -> 90 to be certain)
        unsigned int txSizeEstimate = 90;

        //find masternode rewards that need to be combined
        CCoinControl* coinControl = new CCoinControl();
        CAmount nTotalRewardsValue = 0;
        for (const COutput& out : vCoins) {
            if (!out.fSpendable)
                continue;
            //no coins should get this far if they dont have proper maturity, this is double checking
            if (out.tx->IsCoinStake() && out.tx->GetDepthInMainChain() < Params().COINBASE_MATURITY() + 1)
                continue;

            COutPoint outpt(out.tx->GetHash(), out.i);
            coinControl->Select(outpt);
            vRewardCoins.push_back(out);
            nTotalRewardsValue += out.Value();

            // Combine to the threshold and not way above
            if (nTotalRewardsValue > nAutoCombineThreshold * COIN)
                break;

            // Around 180 bytes per input. We use 190 to be certain
            txSizeEstimate += 190;
            if (txSizeEstimate >= MAX_STANDARD_TX_SIZE - 200) {
                maxSize = true;
                break;
            }
        }

        //if no inputs found then return
        if (!coinControl->HasSelected())
            continue;
```

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

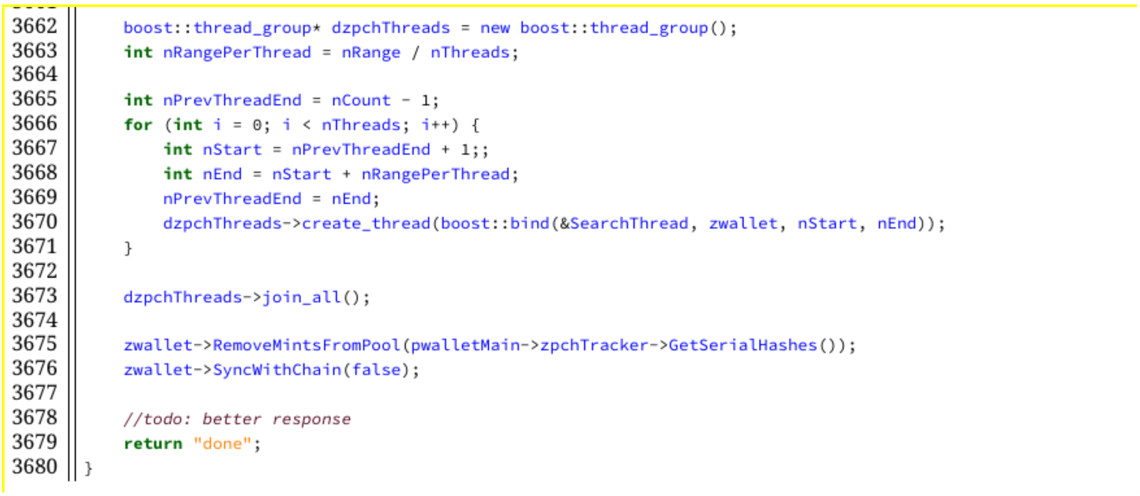
In **src/miner.cpp:CreateNewBlock**, reference to 'publicSpoend' and 'spendObj' member variable is used after 'publicSpend' and 'spendObj' is freed from the stack. It can be led to data corruption.



```
        // double check that there are no double spent zPCH spends in this block or tx
        if (tx.HasZerocoinSpendInputs()) {
            int nHeightTx = 0;
            if (IsTransactionInChain(tx.GetHash(), nHeightTx))
                continue;

            bool fDoubleSerial = false;
            for (const CTxIn& txIn : tx.vin) {
                bool isPublicSpend = txIn.IsZerocoinPublicSpend();
                if (txIn.IsZerocoinSpend() || isPublicSpend) {
                    libzerocoin::CoinSpend* spend;
                    if (isPublicSpend) {
                        libzerocoin::ZerocoinParams* params = Params().Zerocoin_Params(false);
                        PublicCoinSpend publicSpend(params);
                        CValidationState state;
                        if (!ZPCHModule::ParseZerocoinPublicSpend(txIn, tx, state, publicSpend)){
                            throw std::runtime_error("Invalid public spend parse");
                        }
                        spend = &publicSpend;
                    } else {
                        libzerocoin::CoinSpend spendObj = TxInToZerocoinSpend(txIn);
                        spend = &spendObj;
                    }

                    bool fUseV1Params = libzerocoin::ExtractVersionFromSerial(spend->
                         getCoinSerialNumber()) < libzerocoin::PrivateCoin::PUBKEY_VERSION;
                    if (!spend->HasValidSerial(Params().Zerocoin_Params(fUseV1Params)))
                        fDoubleSerial = true;
```

## 1.4 CID 242476 - Resource leak

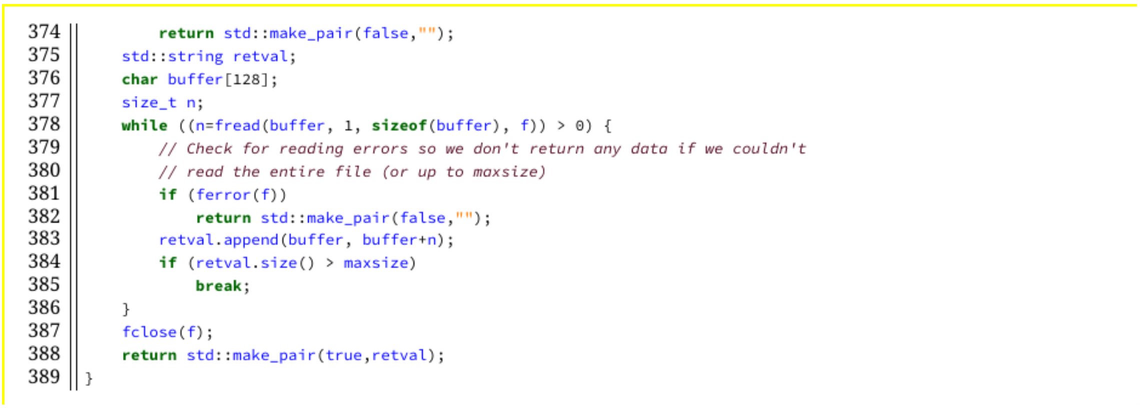
In **src/wallet/rpcwallet.cpp:serchdzpch**, dzpchThreads is dynamically allocated but not freed before control leaves the function. It will cause a memory leak which can disrupt stability.



```
    boost::thread_group* dzpchThreads = new boost::thread_group();
    int nRangePerThread = nRange / nThreads;

    int nPrevThreadEnd = nCount - 1;
    for (int i = 0; i < nThreads; i++) {
        int nStart = nPrevThreadEnd + 1;;
        int nEnd = nStart + nRangePerThread;
        nPrevThreadEnd = nEnd;
        dzpchThreads->create_thread(boost::bind(&SearchThread, zwallet, nStart, nEnd));
    }

    dzpchThreads->join_all();

    zwallet->RemoveMintsFromPool(pwalletMain->zpchTracker->GetSerialHashes());
    zwallet->SyncWithChain(false);

    //todo: better response
    return "done";
}
```

## 1.5 CID 242474 - Resource leak

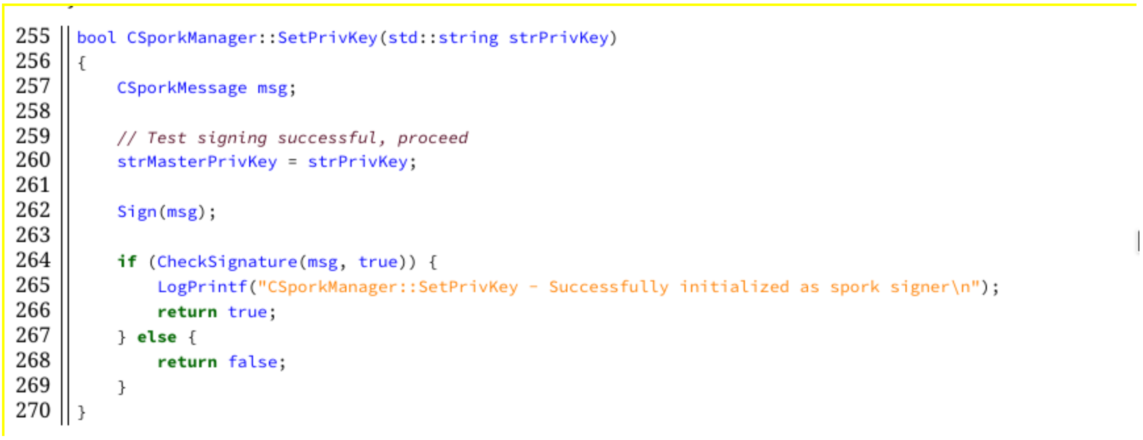
In **src/torcontrol.cpp:ReadBinaryFile**, the file is not properly closed if file descriptor encountered an error. It can cause file descriptor leak which can lead to system instability.



```
        return std::make_pair(false,"");
    std::string retval;
    char buffer[128];
    size_t n;
    while ((n=fread(buffer, 1, sizeof(buffer), f)) > 0) {
        // Check for reading errors so we don't return any data if we couldn't
        // read the entire file (or up to maxsize)
        if (ferror(f))
            return std::make_pair(false,"");
        retval.append(buffer, buffer+n);
        if (retval.size() > maxsize)
            break;
    }
    fclose(f);
    return std::make_pair(true,retval);
}
```

## 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.

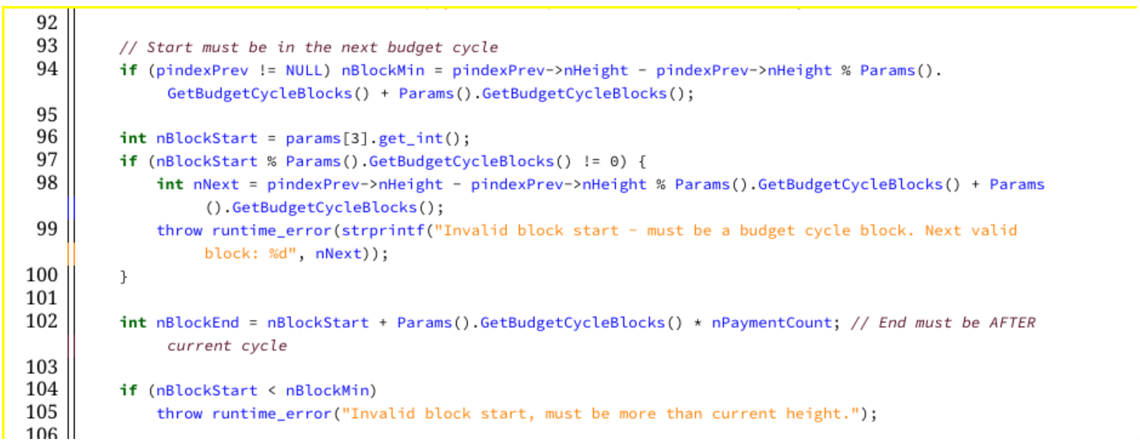


```
bool CSporkManager::SetPrivKey(std::string strPrivKey)
{
    CSporkMessage msg;

    // Test signing successful, proceed
    strMasterPrivKey = strPrivKey;

    Sign(msg);

    if (CheckSignature(msg, true)) {
        LogPrintf("CSporkManager::SetPrivKey - Successfully initialized as spork signer\n");
        return true;
    } else {
        return false;
    }
}
```

## 2 Static analysis using Synopsis® Coverity® Scan - Medium impact issues

### 2.1 CID 242536 - Dereference after null check

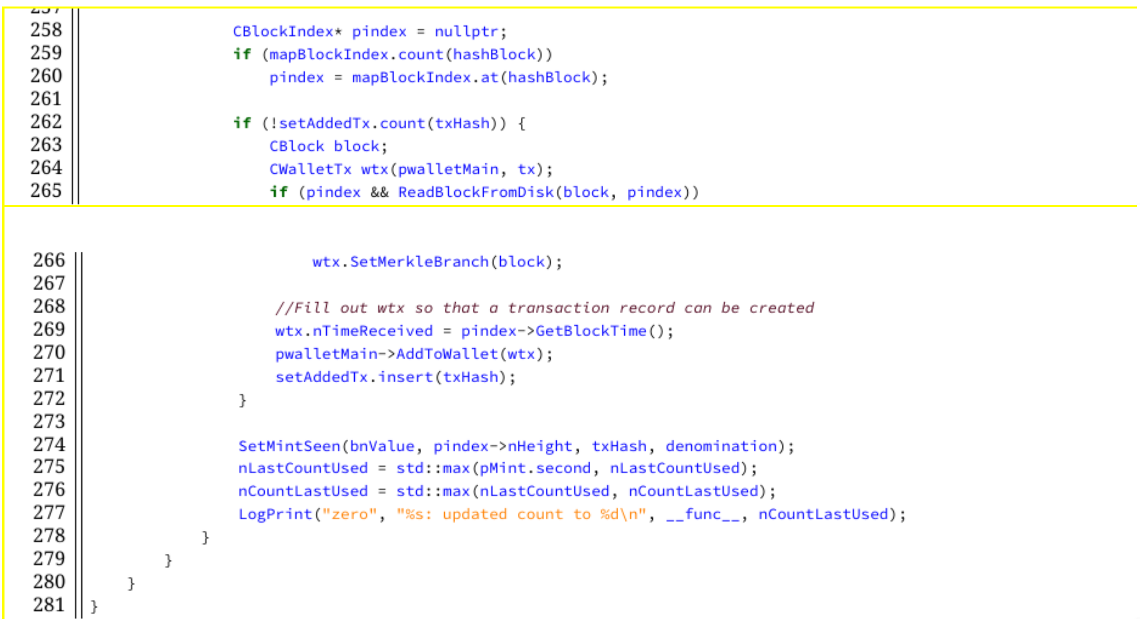
In **src/rpc/budget.cpp:preparebudget**, pindexPrev can be null in a special case. Though it will not cause a practical problem in production due to its special condition requirement, it is good to fix this problem to improve code quality.



```
    // Start must be in the next budget cycle
    if (pindexPrev != NULL) nBlockMin = pindexPrev->nHeight - pindexPrev->nHeight % Params().
        GetBudgetCycleBlocks() + Params().GetBudgetCycleBlocks();

    int nBlockStart = params[3].get_int();
    if (nBlockStart % Params().GetBudgetCycleBlocks() != 0) {
        int nNext = pindexPrev->nHeight - pindexPrev->nHeight % Params().GetBudgetCycleBlocks() + Params
            ().GetBudgetCycleBlocks();
        throw runtime_error(strprintf("Invalid block start - must be a budget cycle block. Next valid
            block: %d", nNext));
    }

    int nBlockEnd = nBlockStart + Params().GetBudgetCycleBlocks() * nPaymentCount; // End must be AFTER
        current cycle

    if (nBlockStart < nBlockMin)
        throw runtime_error("Invalid block start, must be more than current height.");
```

### 2.2 CID 242518 - Dereference after null check

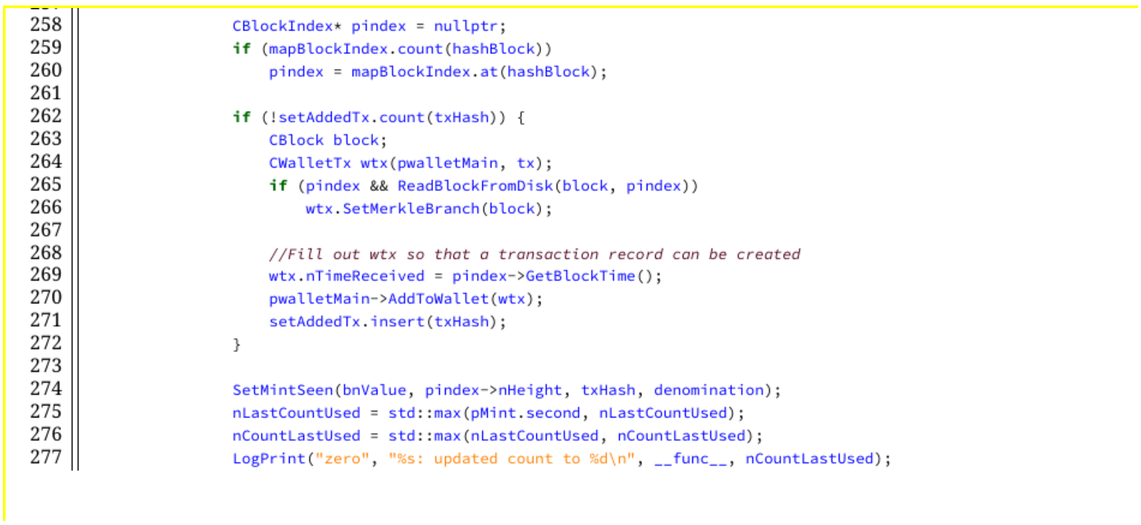
In **src/zpch/zpchwallet.cpp:CzPCHWallet::SyncWithChain**, in case of pindex is null (some special condition is required), it can cause wallet crash.



```
                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);
            }
        }
    }
}
```

## 2.3 CID 242507 - Explicit null dereferenced

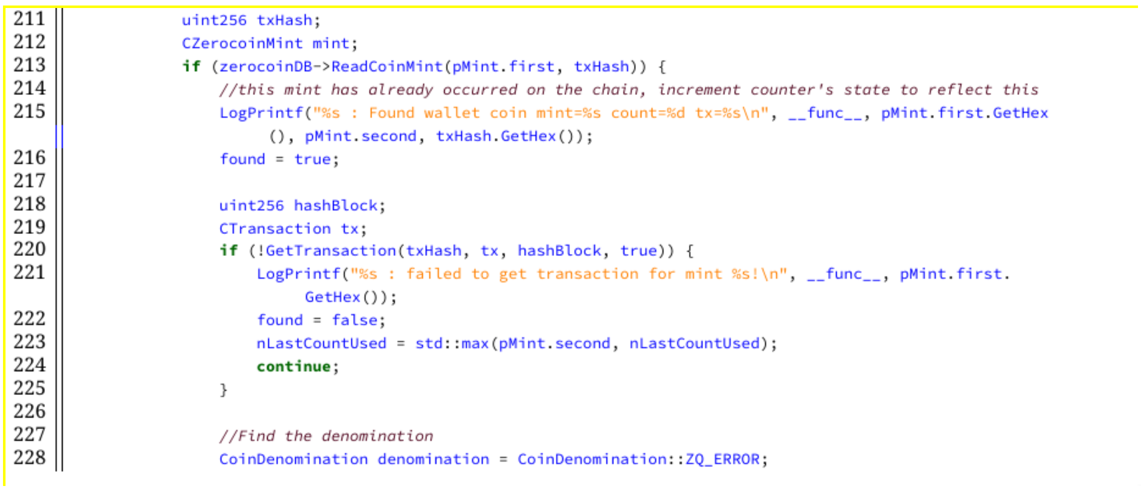
In **src/zpch/zpchwallet.cpp:CzPCHWallet::SyncWithChain**, in special case if mapBlockIndex.count(hashBlock) is zero, pindex will be null. In this case, it will lead to wallet crash.



```
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);
```

## 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.



```
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;
```

```

                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);
            }
        }
    }
}
```

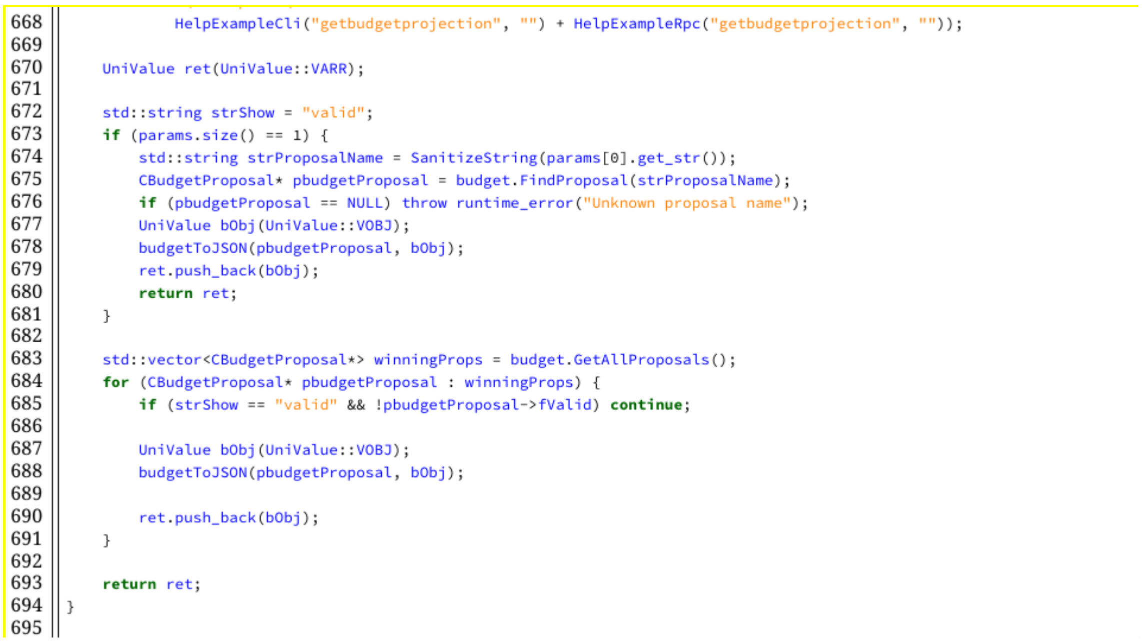
## 2.5 CID 242494 - Dereference after null check

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

```
// Check these inputs the same way we check the vote commands:
// **********************************************************

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.");

int nPaymentCount = params[2].get_int();
if (nPaymentCount < 1)
    throw runtime_error("Invalid payment count, must be more than zero.");

// Start must be in the next budget cycle
if (pindexPrev != NULL) nBlockMin = pindexPrev->nHeight - pindexPrev->nHeight % Params().
    GetBudgetCycleBlocks() + Params().GetBudgetCycleBlocks();

int nBlockStart = params[3].get_int();
if (nBlockStart % Params().GetBudgetCycleBlocks() != 0) {
    int nNext = pindexPrev->nHeight - pindexPrev->nHeight % Params().GetBudgetCycleBlocks() + Params
        ().GetBudgetCycleBlocks();
    throw runtime_error(strprintf("Invalid block start - must be a budget cycle block. Next valid
        block: %d", nNext));
}

int nBlockEnd = nBlockStart + (Params().GetBudgetCycleBlocks() * nPaymentCount); // End must be AFTER
    current cycle

if (nBlockStart < nBlockMin)
    throw runtime_error("Invalid block start, must be more than current height.");

    if (nBlockEnd < pindexPrev->nHeight)
        throw runtime_error("Invalid ending block, starting block + (payment_cycle*payments) must be more
              than current height.");

    CBitcoinAddress address(params[4].get_str());
    if (!address.IsValid())
        throw JSONRPCError(RPC_INVALID_ADDRESS_OR_KEY, "Invalid PIVX address");

    // Parse PIVX address
    CScript scriptPubKey = GetScriptForDestination(address.Get());
    CAmount nAmount = AmountFromValue(params[5]);
    uint256 hash = ParseHashV(params[6], "parameter 1");

    //create the proposal incase we're the first to make it
    CBudgetProposalBroadcast budgetProposalBroadcast(strProposalName, strURL, nPaymentCount, scriptPubKey
         , nAmount, nBlockStart, hash);

    std::string strError = "";
    int nConf = 0;
    if (!IsBudgetCollateralValid(hash, budgetProposalBroadcast.GetHash(), strError,
         budgetProposalBroadcast.nTime, nConf)) {
        throw runtime_error("Proposal FeeTX is not valid - " + hash.ToString() + " - " + strError);
    }

    if (!masternodeSync.IsBlockchainSynced()) {
        throw runtime_error("Must wait for client to sync with masternode network. Try again in a minute
             or so.");
    }

    // if(!budgetProposalBroadcast.IsValid(strError)){
    //     return "Proposal is not valid - " + budgetProposalBroadcast.GetHash().ToString() + " - " +
         strError;
    // }

    budget.mapSeenMasternodeBudgetProposals.insert(make_pair(budgetProposalBroadcast.GetHash(),
         budgetProposalBroadcast));
    budgetProposalBroadcast.Relay();
    if(budget.AddProposal(budgetProposalBroadcast)) {
        return budgetProposalBroadcast.GetHash().ToString();
    }
    throw runtime_error("Invalid proposal, see debug.log for details.");
}
```

## 2.6 CID 242485 - Dereference null return value

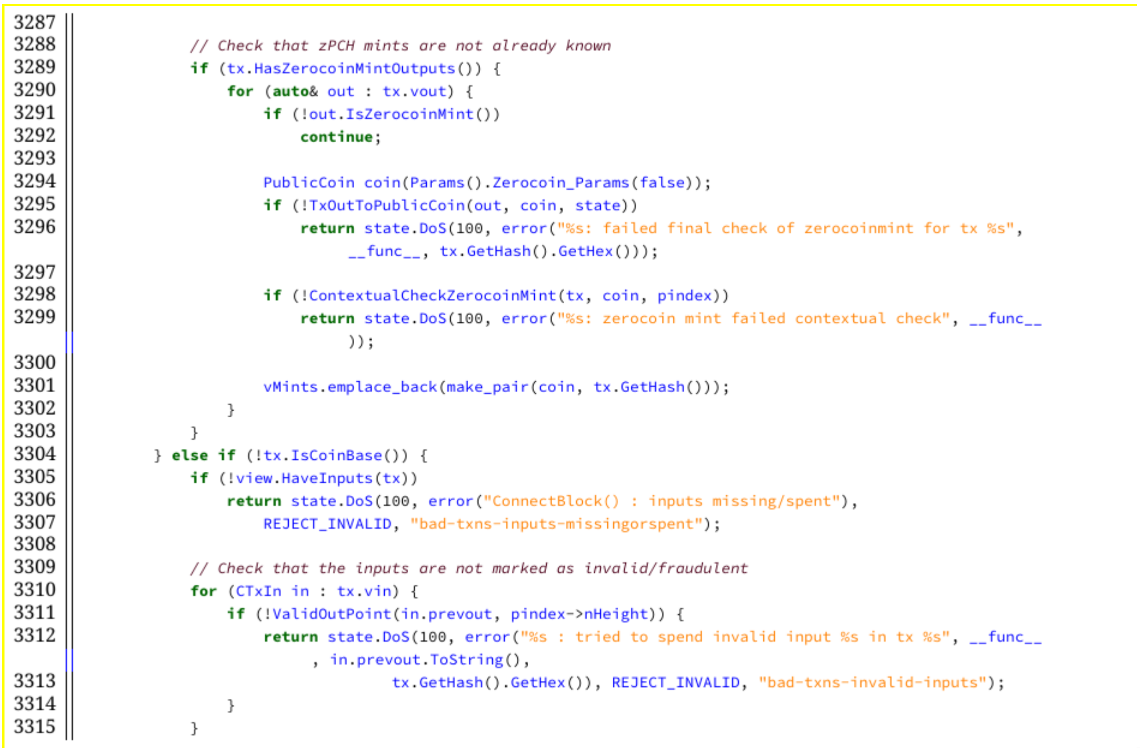
In **src/rpc/rawtransaction.cpp:signrawtransaction**, when wallet is in regression test mode, there is possibility to crash wallet with crafted request. This vulnerability will not affect production environment because regression test cannot be connected to internet but it is good to fix this problem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



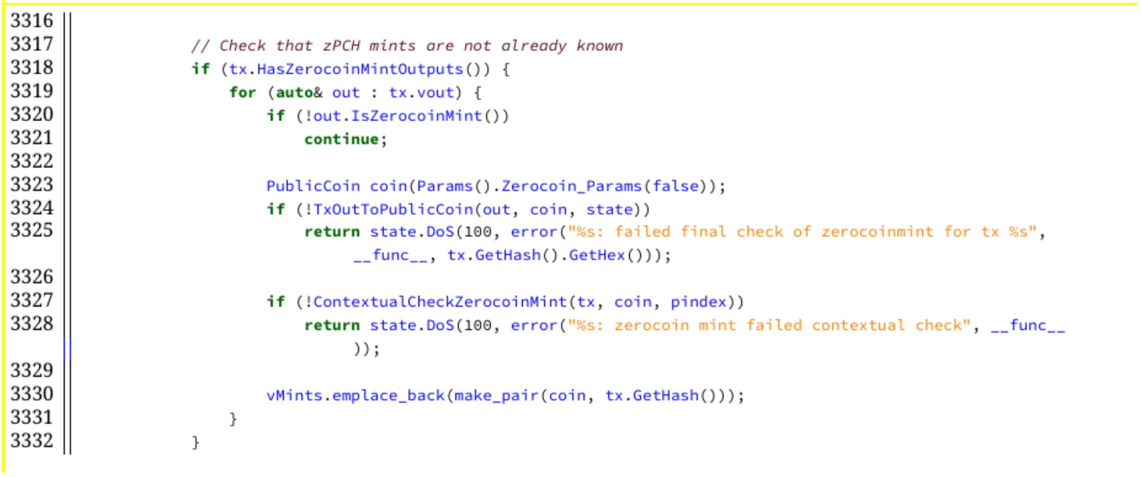
```
            HelpExampleCli("getbudgetprojection", "") + HelpExampleRpc("getbudgetprojection", ""));

    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;
}
```

## 2.7 CID 242477 - Dereference after null check

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



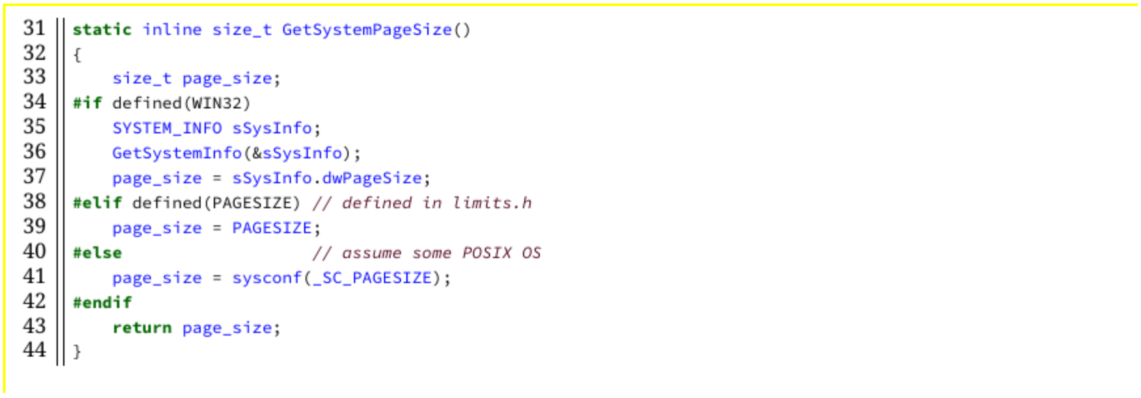
```
            // Check that zPCH mints are not already known
            if (tx.HasZerocoinMintOutputs()) {
                for (auto& out : tx.vout) {
                    if (!out.IsZerocoinMint())
                        continue;

                    PublicCoin coin(Params().Zerocoin_Params(false));
                    if (!TxOutToPublicCoin(out, coin, state))
                        return state.DoS(100, error("%s: failed final check of zerocoinmint for tx %s",
                             __func__, tx.GetHash().GetHex()));

                    if (!ContextualCheckZerocoinMint(tx, coin, pindex))
                        return state.DoS(100, error("%s: zerocoin mint failed contextual check", __func__
                             ));

                    vMints.emplace_back(make_pair(coin, tx.GetHash()));
                }
            }
        } else if (!tx.IsCoinBase()) {
            if (!view.HaveInputs(tx))
                return state.DoS(100, error("ConnectBlock() : inputs missing/spent"),
                    REJECT_INVALID, "bad-txns-inputs-missingorspent");

            // Check that the inputs are not marked as invalid/fraudulent
            for (CTxIn in : tx.vin) {
                if (!ValidOutPoint(in.prevout, pindex->nHeight)) {
                    return state.DoS(100, error("%s : tried to spend invalid input %s in tx %s", __func__
                         , in.prevout.ToString(),
                                  tx.GetHash().GetHex()), REJECT_INVALID, "bad-txns-invalid-inputs");
                }
            }
```



```
            // Check that zPCH mints are not already known
            if (tx.HasZerocoinMintOutputs()) {
                for (auto& out : tx.vout) {
                    if (!out.IsZerocoinMint())
                        continue;

                    PublicCoin coin(Params().Zerocoin_Params(false));
                    if (!TxOutToPublicCoin(out, coin, state))
                        return state.DoS(100, error("%s: failed final check of zerocoinmint for tx %s",
                             __func__, tx.GetHash().GetHex()));

                    if (!ContextualCheckZerocoinMint(tx, coin, pindex))
                        return state.DoS(100, error("%s: zerocoin mint failed contextual check", __func__
                             ));

                    vMints.emplace_back(make_pair(coin, tx.GetHash()));
                }
            }
```

## 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.



```
static inline size_t GetSystemPageSize()
{
    size_t page_size;
#if defined(WIN32)
    SYSTEM_INFO sSysInfo;
    GetSystemInfo(&sSysInfo);
    page_size = sSysInfo.dwPageSize;
#elif defined(PAGESIZE) // defined in limits.h
    page_size = PAGESIZE;
#else                   // assume some POSIX OS
    page_size = sysconf(_SC_PAGESIZE);
#endif
    return page_size;
}
```

## Conclusions

Our research team found some issues on POPCHAIN core code, but those issues had been fixed with this report. **POPCHAIN mainnet would fully work for purpose.** This is guaranteed by SCV SOFT vulnerability research team.

This audit report was performed by **SCV SOFT**

**Benjamin Hyokeun Oh,** CEO

August 21st, 2019

[illegible] 代表理事