Smart Contract

The audit was conducted by TECHFUND between 17th to 26th of October. The code used was from master branch available on 17th October via following repositories

https://github.com/quras-official/quras-blockchain-csharp
https://github.com/quras-official/quras-smartcontract-compiler

Vulnerabilities

High	1
Medium	5
Low	1
Note	2

We found above vulnerabilities in the code that have been described below. We also tried to perform flooding the network, but were unable to achieve any Critical Issue in the same.

1) **Issue : String comparison Incorrect OPCODE generation**
HIGH

```
using Quras.SmartContract.Framework;
using Quras.SmartContract.Framework.Services.Module;
using System;
using System.Numerics;

namespace qtest
{
    0 references
    public class Contract1 : SmartContract
    {
        0 references
        public static bool Main(string operation, object[] args)
        {
            if (operation == "test") Runtime.Log("This is test");
            if (operation != "nottest") Runtime.Log("This is not test");
            return true;
        }
    }
}
```

Generated OPCODES for first condition :

```
c3 // PICKITEM
04 // ??
74 // t
65 // e
73 // s
74 // t
87 // equal
6c // FROMALTSTACK
76 // dup
6b // TOALTSTACK
52 // PUSH2
52 // PUSH2
7a // roll
c4 // setitem
6c // FROMALTSTACK
76 // dup
6b // TOALTSTACK
52 // PUSH2
c3 // PICKITEM
64 // JMPIFNOT
25 // 37
00 //
0c // ?
54 T
68 h
69 i
73 s
20
69 i
73 s
20
74 t
65 e
73 s
74 t
61 // NOP
68 //syscall
```

The jump here is correct to FORMALTSTACK for comparison with first operation

Generated OPCODES for second condition :

```
c3
64 // JMPIFNOT
29
00
10
54 T
68 h
69 i
73 s
20
69 i
73 s
20
6e n
6f o
74 t
20
74 t
65 e
73 s
74 t
61 // NOP
```

JMPIFNOT jumps to a location which is not valid.

Hence the output is as follows :

Expected output for operation="test" :
"This is test"
"This is not test"

Output :
"This is test"

Reason :

Invalid else condition (https://github.com/quras-official/quras-smartcontract-compiler/blob/master/quras_msil/MSIL/Conv_Multi.cs#L508)

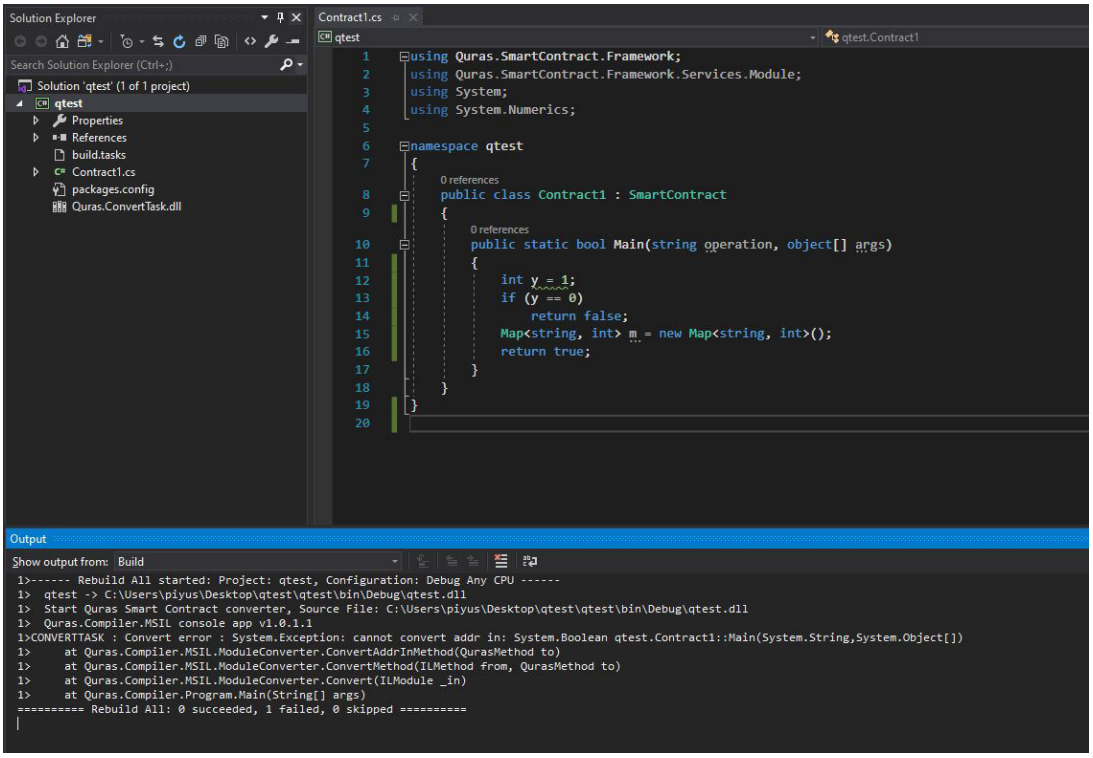
_Convert1by1(Quras.VM.OpCode.INVERT, src, to);
_Insert1(Quras.VM.OpCode.EQUAL, "", to);

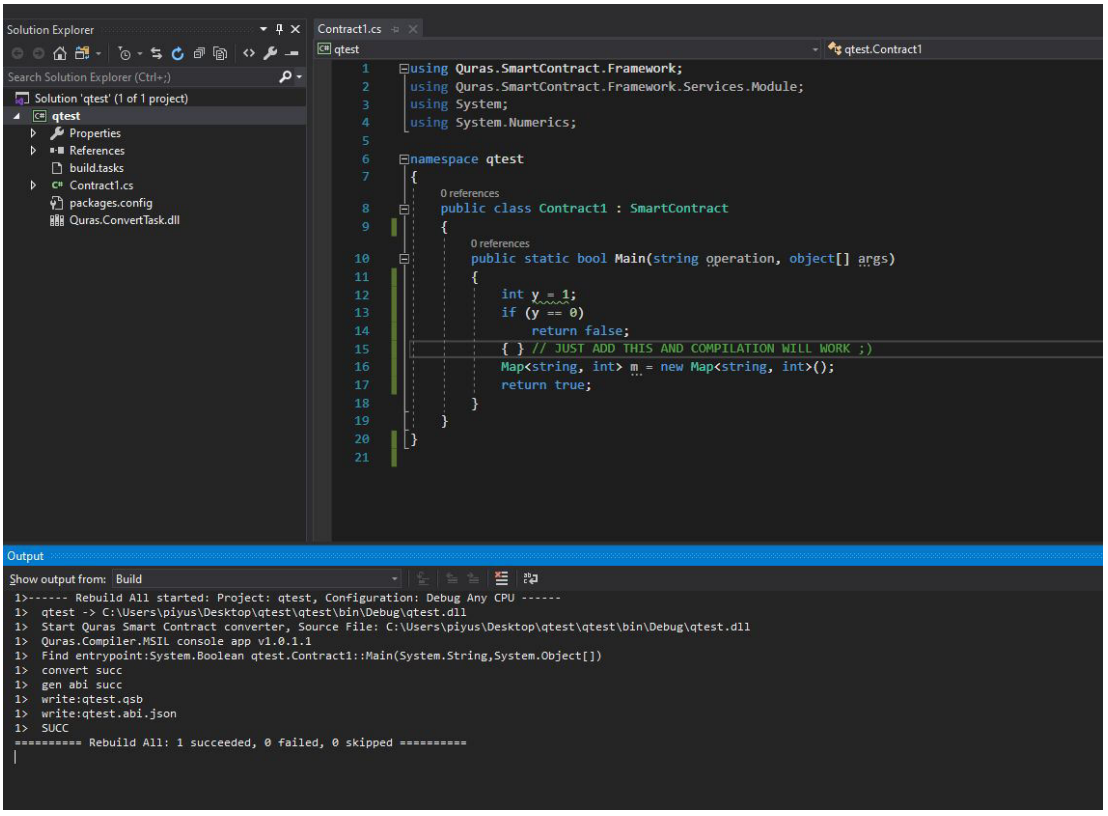
Should be changed to :

_Convert1by1(Quras.VM.OpCode.EQUAL, src, to);
_Convert1by1(Quras.VM.OpCode.NOT, null, to);

```
496                 }
497                 else if (src.tokenMethod.Contains("::op_Inequality("))
498                 {
499                     var _ref = src.tokenUnknown as Mono.Cecil.MethodReference;
500                     if (_ref.DeclaringType.FullName == "System.Boolean"
501                         || _ref.DeclaringType.FullName == "System.Int32"
502                         || _ref.DeclaringType.FullName == "System.Numerics.BigInteger")
503                     {
504                         _Convert1by1(Quras.VM.OpCode.NUMNOTEQUAL, src, to);
505                     }
506                     else
507                     {
508                         _Convert1by1(Quras.VM.OpCode.INVERT, src, to);
509                         _Insert1(Quras.VM.OpCode.EQUAL, "", to);
510                     }
```

2) **Compilation fails for valid code** Medium





Reason :
https://github.com/quras-official/quras-smartcontract-compiler/blob/master/quras_msil/MSIL/Conv_Multi.cs#L1081

3) **Variables in Contract creates issue** Low

```
namespace qtest
{
    1 reference
    public class Contract1 : SmartContract
    {
        public static string randomstring = "thisis atest111!!";
        0 references
        public static void Main(string operation, params object[] args)
        {
            Contract2.TestString();
        }
    }

    1 reference
    public class Contract2 : SmartContract
    {
        1 reference
        public static void TestString()
        {
            Runtime.Log(Contract1.randomstring);
        }
    }
}
```

Variables that are not `readonly` **fail silently** without throwing any exception leading to unexpected output.

4) **Incorrect SETITEM** Medium

```
namespace qtest
{
    0 references
    public class Contract1 : SmartContract
    {
        0 references
        public static byte[] Main()
        {
            byte[] b = new byte[] { 0x02, 0x02, 0x02, 0x02, 0x02 };
            b[2] = 0x05;
            return b;
        }
    }
}
```

The above Smart Contract leads to following OPCODES

```
1   52
2
3   c5 // new array
4   6b // TOALTSTACK
5   61 // nop
6
7   05
8   02 // DATA
9   02 // DATA
10  02 // DATA
11  02 // DATA
12  02 // DATA
13  6c // FROMALTSTACK
14  76 // DUP
15  6b // TOALTSTACK
16
17  00 // push0
18  52 // push2
19  7a // ROLL
20  c4 // SETITEM
21  6c // FROMALTSTACK
22  76 // dup
23  6b // TOALTSTACK
24  00 // push0
25  c3 // pickitem
26
27  52 // push2
28  55 // push5
29  c4 // setitem
30  6c // FROMALTSTACK
31
```

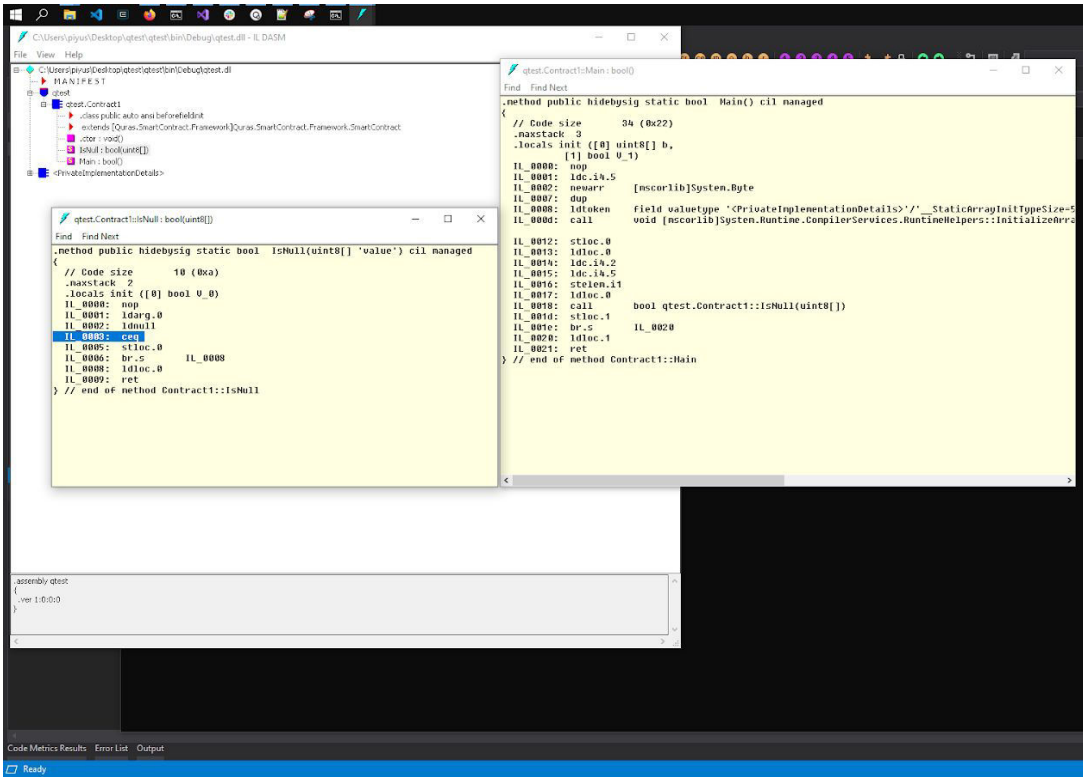
Here in line 29 SETITEM fails as it is allowed for arrays or maps (not bytearray).

5) NULL : The checks doesn't work as expected

Medium

```
namespace qtest
{
    0 references
    public class Contract1 : SmartContract
    {
        1 reference
        public static bool IsNull(byte[] value)
        {
            return value == null;
        }

        0 references
        public static bool Main()
        {
            byte[] b = new byte[] { 0x02, 0x02, 0x02, 0x02, 0x02 };
            b[2] = 0x05;
            return IsNull(b);
        }
    }
}
```

The above code gives following ILCode



But ceq is defined by NUMEQUAL

```
749                 case CodeEx.Ceq:
750                     _Convert1by1(Quras.VM.OpCode.NUMEQUAL, src, to);
751                     break;
752
```

VM needs to accept 'null' too in NUMEQUAL.

6) Issue : The contract should get destroyed after the migration is complete, but the function just returns a boolean value. Here return must be replaced by : Contract_Destroy(engine); NOTE

https://github.com/quras-official/quras-blockchain-csharp/blob/master/QurasCore/SmartContract/StateMachine.cs

```
        ContractState contract = contracts.TryGet(hash);
        if (contract == null)
        {
            contract = new ContractState
            {
                Script = script,
                ParameterList = parameter_list,
                ReturnType = return_type,
                HasStorage = need_storage,
                Name = name,
                CodeVersion = version,
                Author = author,
                Email = email,
                Description = description
            };
            contracts.Add(hash, contract);
            contracts_created.Add(hash, new UInt160(engine.CurrentContext.ScriptHash));
            if (need_storage)
            {
                foreach (var pair in storages.Find(engine.CurrentContext.ScriptHash).ToArray())
                {
                    storages.Add(new StorageKey
                    {
                        ScriptHash = hash,
                        Key = pair.Key.Key
                    }, new StorageItem
                    {
                        Value = pair.Value.Value
                    });
                }
            }
        }
        engine.EvaluationStack.Push(StackItem.FromInterface(contract));
        return true;
    }
```

7) Issue : Resource leakage during persisting block in Level DB **Medium**

https://github.com/quras-official/quras-blockchain-csharp/blob/master/QurasCore/Implementations/Blockchains/LevelDB/LevelDBBlockchain.cs

```
                case TransactionType.PublishTransaction:
                    {
pragma warning disable CS0612
                        PublishTransaction publish_tx = (PublishTransaction)tx;
                        contracts.GetOrAdd(publish_tx.ScriptHash, () => new ContractState
                        {
                            Script = publish_tx.Script,
                            ParameterList = publish_tx.ParameterList,
                            ReturnType = publish_tx.ReturnType,
                            HasStorage = publish_tx.NeedStorage,
                            Name = publish_tx.Name,
                            CodeVersion = publish_tx.CodeVersion,
                            Author = publish_tx.Author,
                            Email = publish_tx.Email,
                            Description = publish_tx.Description
                        });
pragma warning restore CS0612
                    }
                    break;
                case TransactionType.InvocationTransaction:
                    {
                        InvocationTransaction itx = (InvocationTransaction)tx;
                        CachedScriptTable script_table = new CachedScriptTable(contracts);
                        StateMachine service = new StateMachine(accounts, validators, assets, contracts, storages);
                        ApplicationEngine engine = new ApplicationEngine(TriggerType.Application, itx, script_table, service, itx.Gas);
                        engine.LoadScript(itx.Script, false);
                        if (engine.Execute())
                        {
                            service.Commit();
                            notifications.AddRange(service.Notifications);
                        }
                    }
                    break;
            }
        }
```

8) Issue : Prevent Leakage in application engine during running Smart Contract. **Medium**

https://github.com/quras-official/quras-blockchain-csharp/blob/master/QurasCore/SmartContract/ApplicationEngine.cs

The StateMachine service should be created by “using” statement for the “new” construct. In order to prevent resource leakage for the interop service.

```
public static ApplicationEngine Run(byte[] script, IScriptContainer container = null)
{
    DataCache<UInt160, AccountState> accounts = Blockchain.Default.CreateCache<UInt160, AccountState>();
    DataCache<ECPoint, ValidatorState> validators = Blockchain.Default.CreateCache<ECPoint, ValidatorState>();
    DataCache<UInt256, AssetState> assets = Blockchain.Default.CreateCache<UInt256, AssetState>();
    DataCache<UInt160, ContractState> contracts = Blockchain.Default.CreateCache<UInt160, ContractState>();
    DataCache<StorageKey, StorageItem> storages = Blockchain.Default.CreateCache<StorageKey, StorageItem>();
    CachedScriptTable script_table = new CachedScriptTable(contracts);
    StateMachine service = new StateMachine(accounts, validators, assets, contracts, storages);
    ApplicationEngine engine = new ApplicationEngine(TriggerType.Application, container, script_table, service, Fixed8.Zero, true);
    engine.LoadScript(script, false);
    engine.Execute();
    return engine;
}
```

Web Wallet

The audit was conducted by TECHFUND in the first week of November. The code used was made available as a zip file via a private communication channel.

Vulnerabilities

High	1
Medium	1
Low	2
Note	1

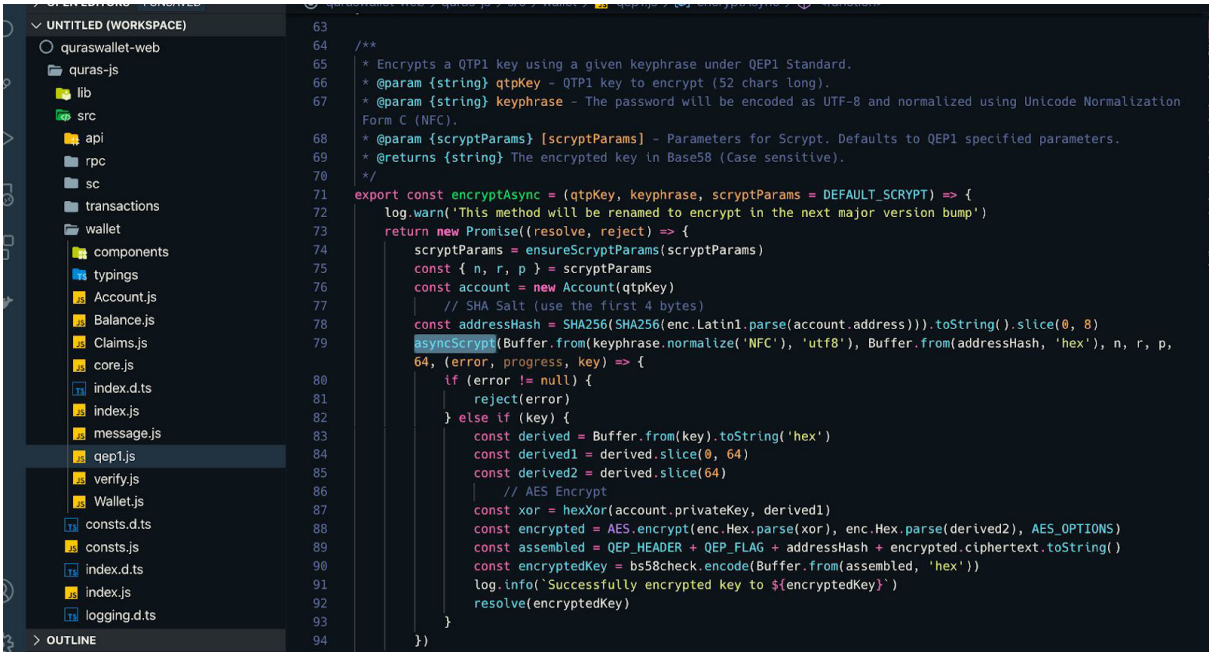
We found above vulnerabilities in the code that have been described below.

1. **Error on passwords with more than 64 bytes**
   HIGH

Quras-js > wallet > Wallet

https://github.com/ricmoo/scrypt-js/issues/11

There is a major issue in the encryption library used,
“When password is more than 64bytes, PBKDF2_HMAC_SHA256_OneIter function runs SHA256 for password.SHA256 expects that the argument is Array but the user uses Buffer.”



The project depends on an external library for scrypt, although this is not an issue on its own, we would highly recommend to use the inbuilt nodejs Crypto module for scrypt implementation. Or Atleast update it to the latest version as the version used is now depreciated.

2. **Issue in some decodings of Fixed8 String**
   MEDIUM

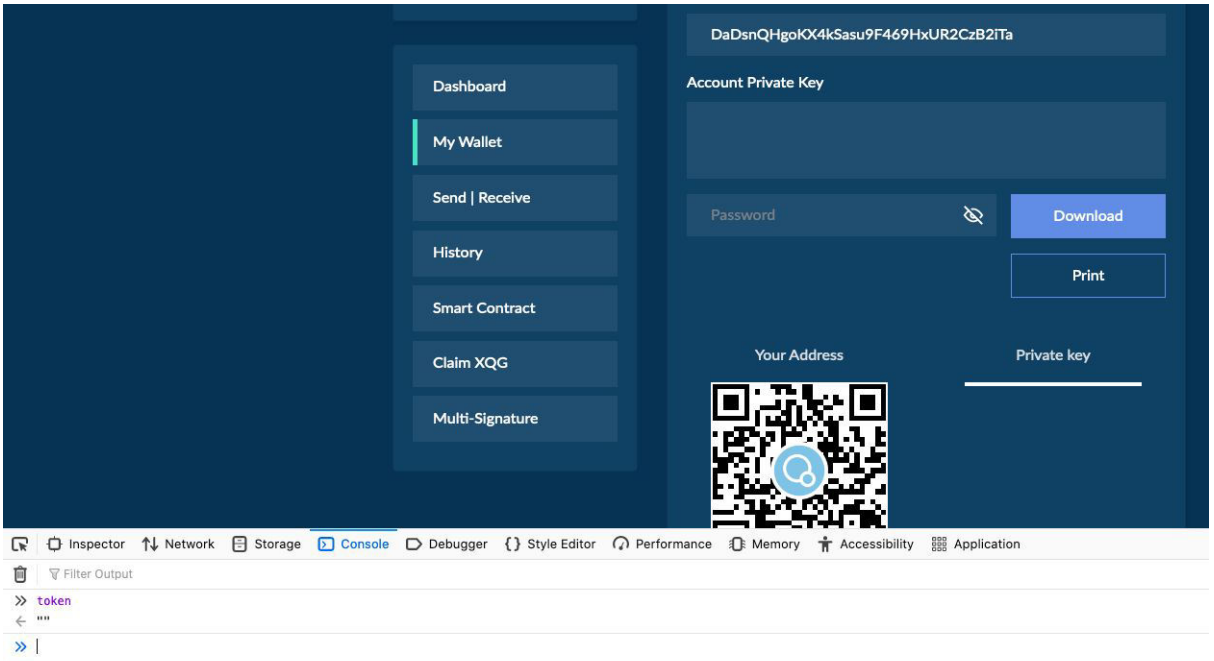
quras-js > src > utils.js

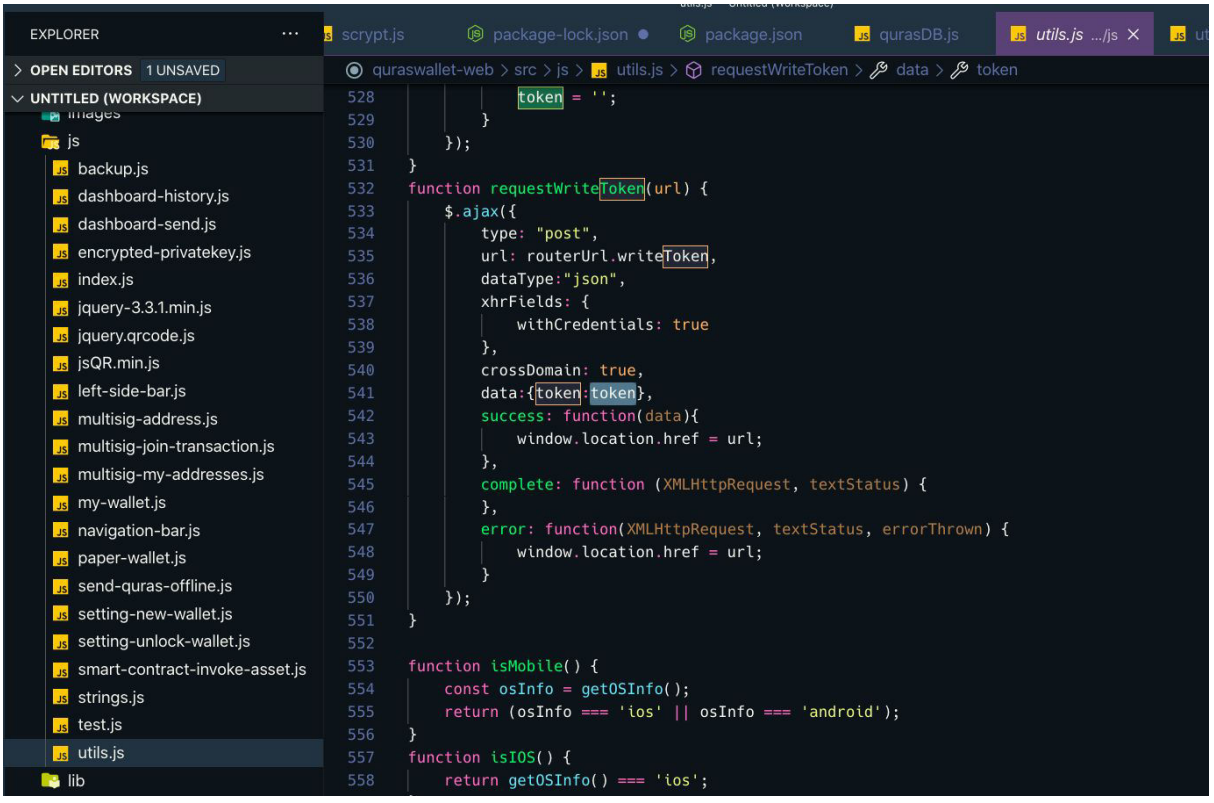
Decoding of fixed8 hex strings will fail for negative input.

Also :
quras.u.fixed82num("ffffffffffffffff") // will fail !

```
export class Fixed8 extends BN {
  constructor (input, base = undefined) {
    var strInput = input.toString()
    var dotIndex = strInput.indexOf('.')
    dotIndex = dotIndex === -1 ? strInput.length - 1 : dotIndex
    input = parseFloat(input).toFixed(strInput.length - dotIndex - 1)
    super(input, base)
  }

  toHex () {
    // In correct !!
    const hexstring = this.times(100000000).round(0).toString(16)
    return '0'.repeat(16 - hexstring.length) + hexstring
  }

  toReverseHex () {
    return reverseHex(this.toHex())
  }

  [util.inspect.custom] (depth, opts) {
    return this.toFixed(8)
  }

  static fromHex (hex) {
    return new Fixed8(hex, 16).div(100000000)
  }

  static fromReverseHex (hex) {
    // In correct !!
    return this.fromHex(reverseHex(hex))
  }
```

3. Token is not set properly and leads to failed API calls LOW



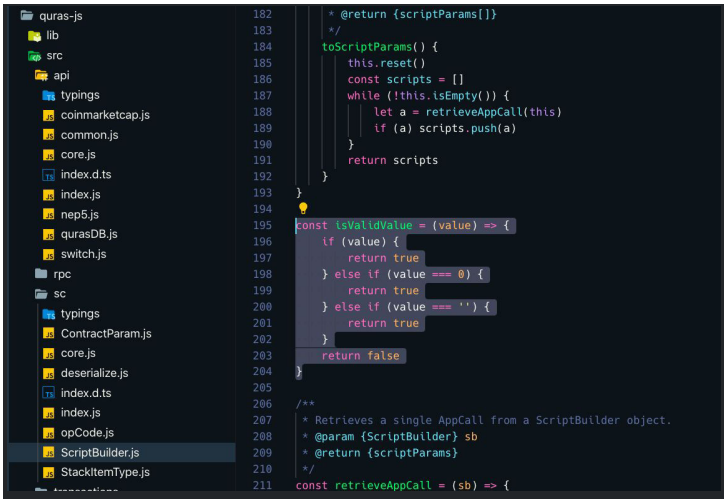


4. False is not a valid implementation in Contract Params
   LOW

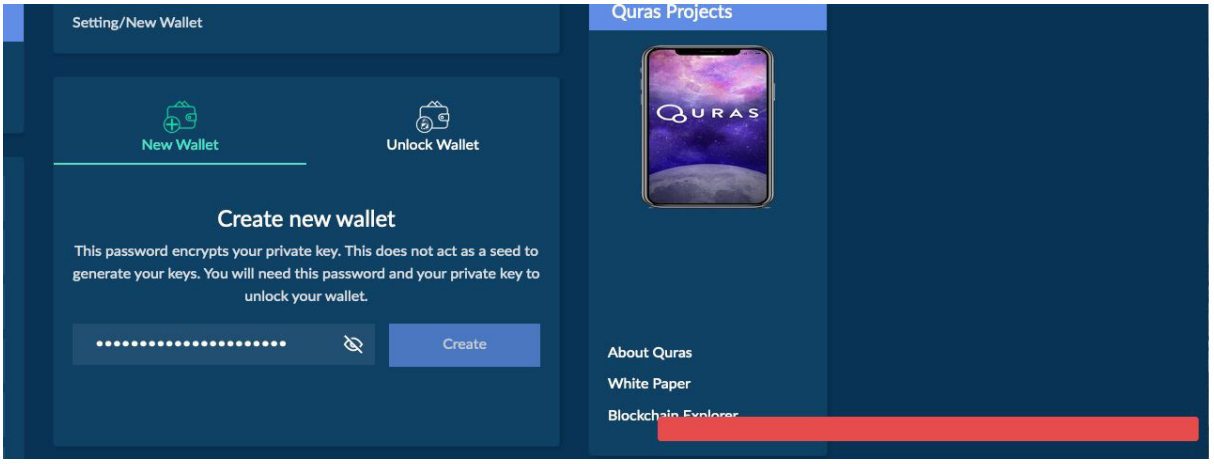
In script builder the false value should be treated as a valid value, currently it is not being handled. Hence if a “false” is passed in Script Builder we get “false” for a correct value.

Solution :

else if(value === false) return true;



5. Create wallet fails for long password inputs without throwing any error
   NOTE



Windows Wallet

The audit was conducted by TECHFUND in November 2020. The code used was from master branch available on 9th November via the following repositories.

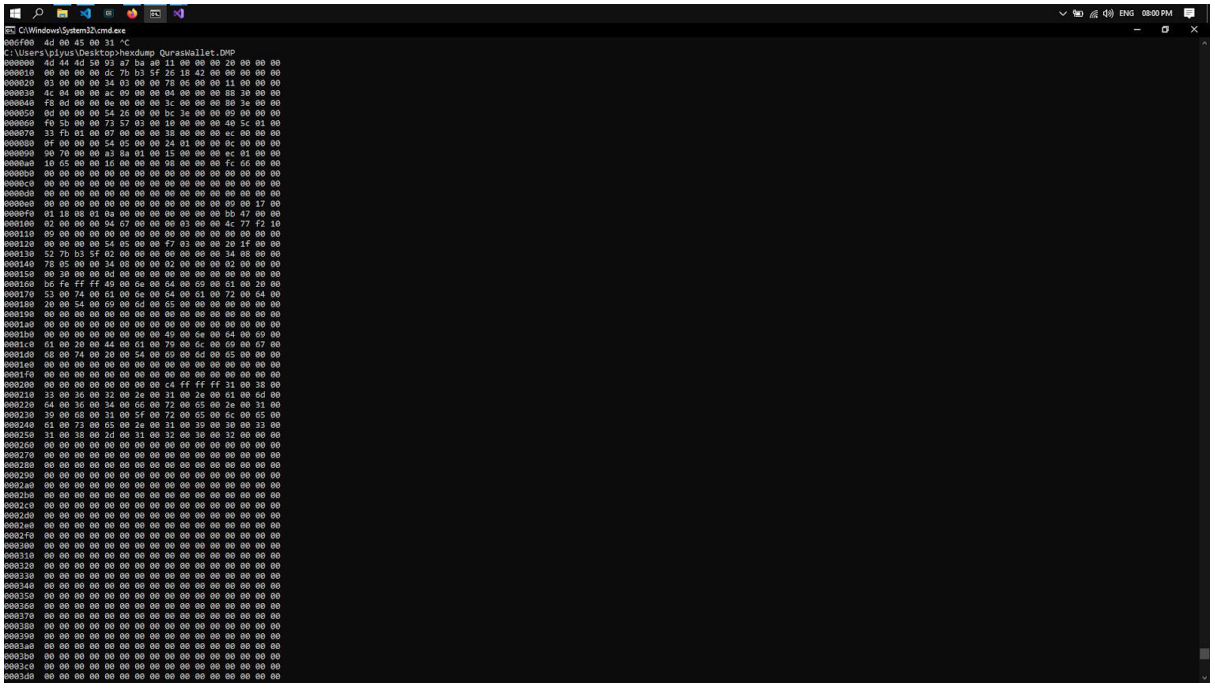
https://bitbucket.org/ros101/quras-core/src/master/
https://github.com/quras-official/smartcontract-nft

Vulnerabilities

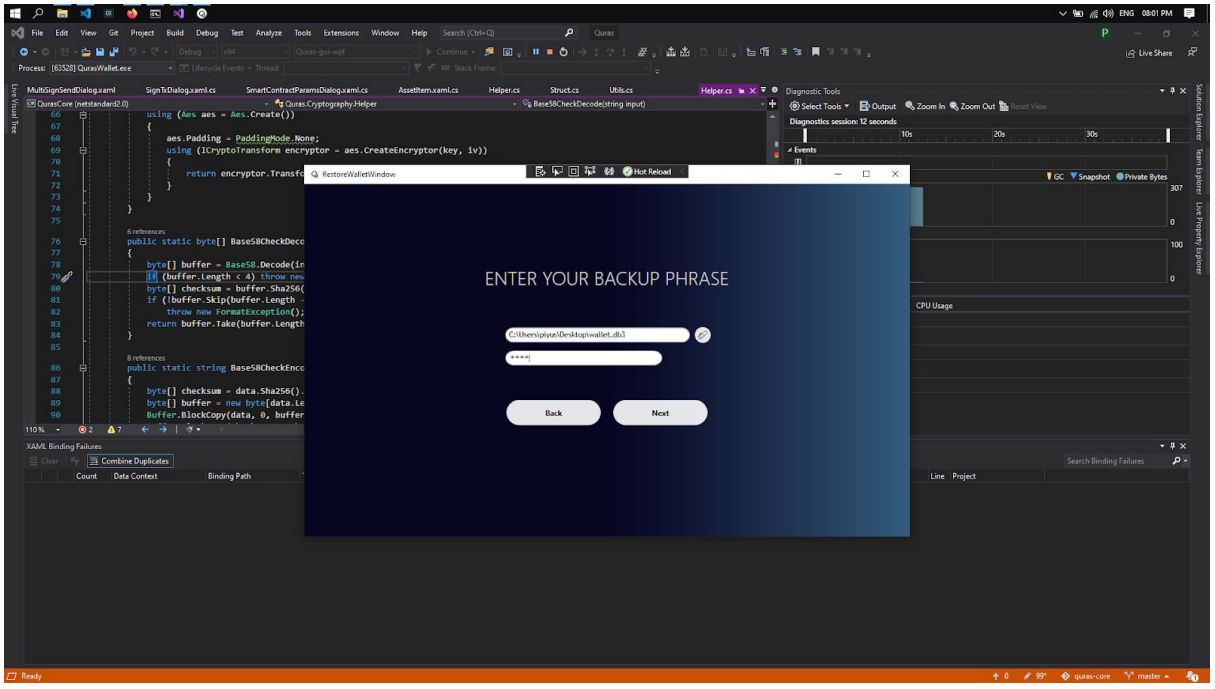
Critical	1
High	1
Medium	3
Low	0
Note	2

We found above vulnerabilities in the code that have been described below.

1) **Issue : Stealing password from running process** CRITICAL



A memory dump was taken while the process was running and user is about to login and is on the following screen



We were successfully able to extract the user password of the wallet from the memory dump. From the memory dump we extracted the strings available inside the dump file.

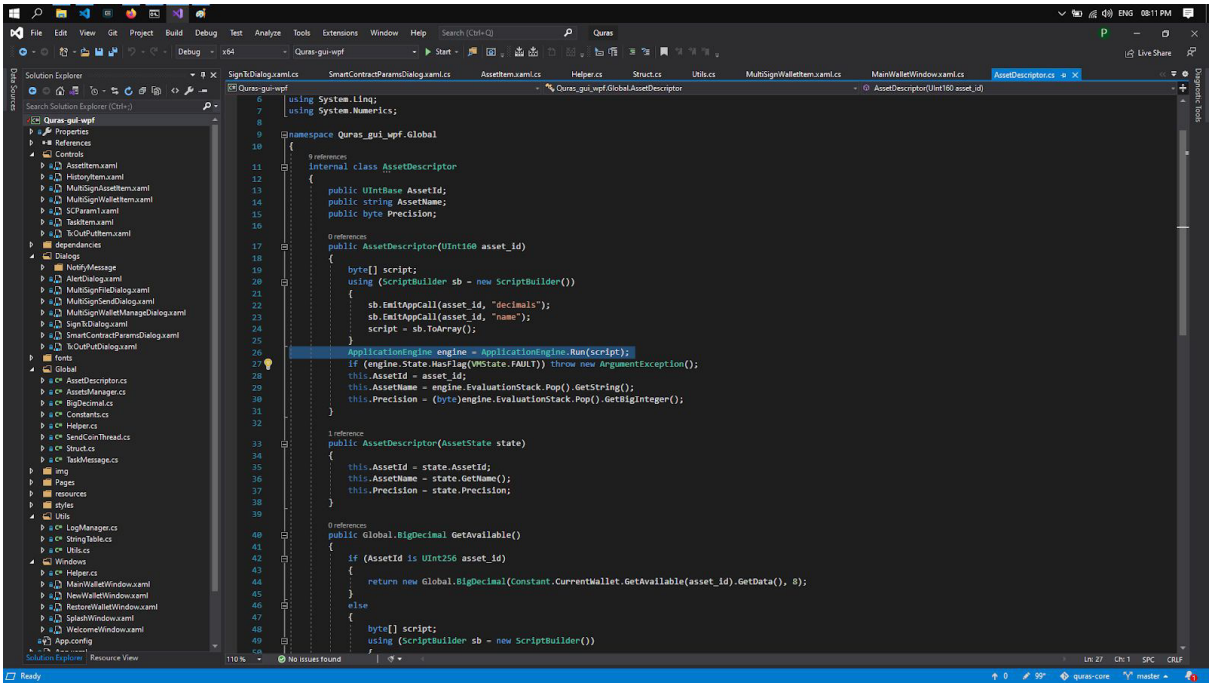
Which allowed us to view the password entered by the user directly. Any third party application on the user system can make use of similar techniques to extract the password. Using similar te techniques it is also possible to grab user addresses and private keys.

```
992262  p'cZ
992263  o?U
992264  8$AU
992265  Q8?
992266  VcZ
992267  VcZ
992268  Q8?
992269  @%F[
992270  Q8?
992271  !AU
992272  !AU
992273  !AU
992274  Q8?
992275  Q8?
992276  ?cZ
992277  8\cZ
992278  HG,[
992279  8\cZ
992280  8\cZ
992281  8\cZ
992282  8\cZ
992283  ?cZ
992284  ?cZ
992285  8\cZ
992286  Password123!
992287  STR_RW_SUCCESS
992288  STR_RW_SUCCESS
992289  ?cZ
992290  STR_RW_SUCCESS
992291  STR_RW_SUCCESS
992292  Language
992293  Language
992294  Language
992295  Language
992296  Password123!
992297  Password123!
992298  0>"u
992299  8?cZ
```

2) Issue : Prevent Leakage in application engine

Medium

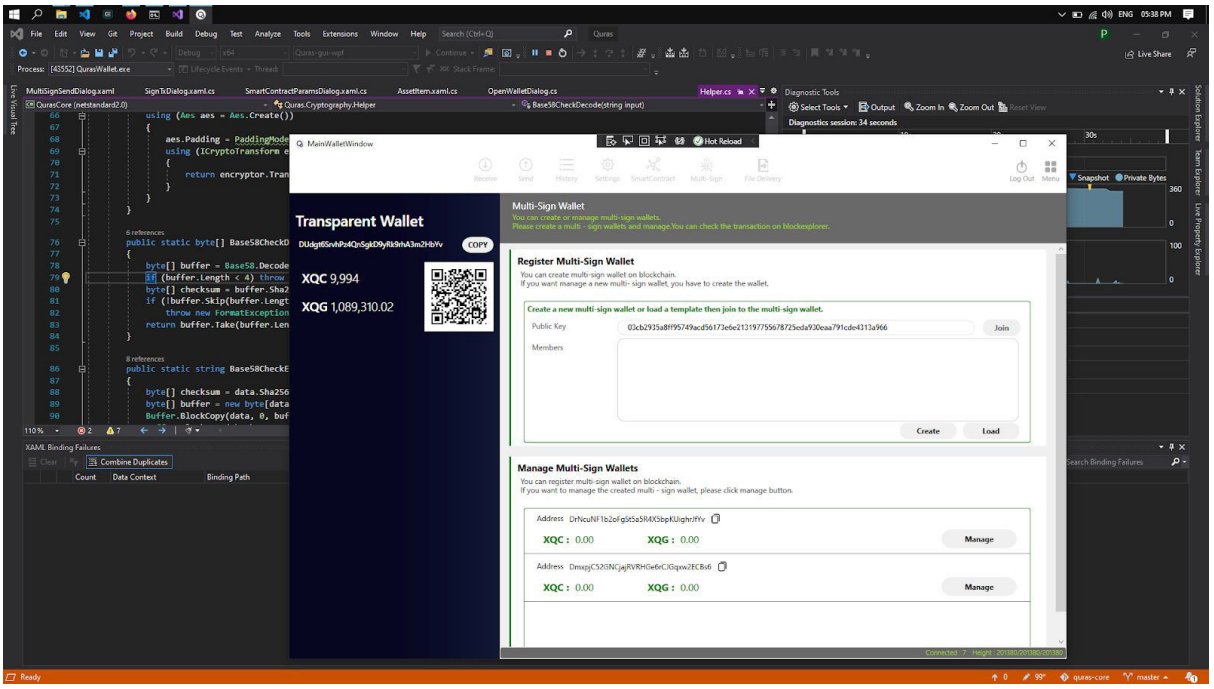
The ApplicationEngine should be wrapped by “using” statement for the “new” construct in order to prevent resource leakage.

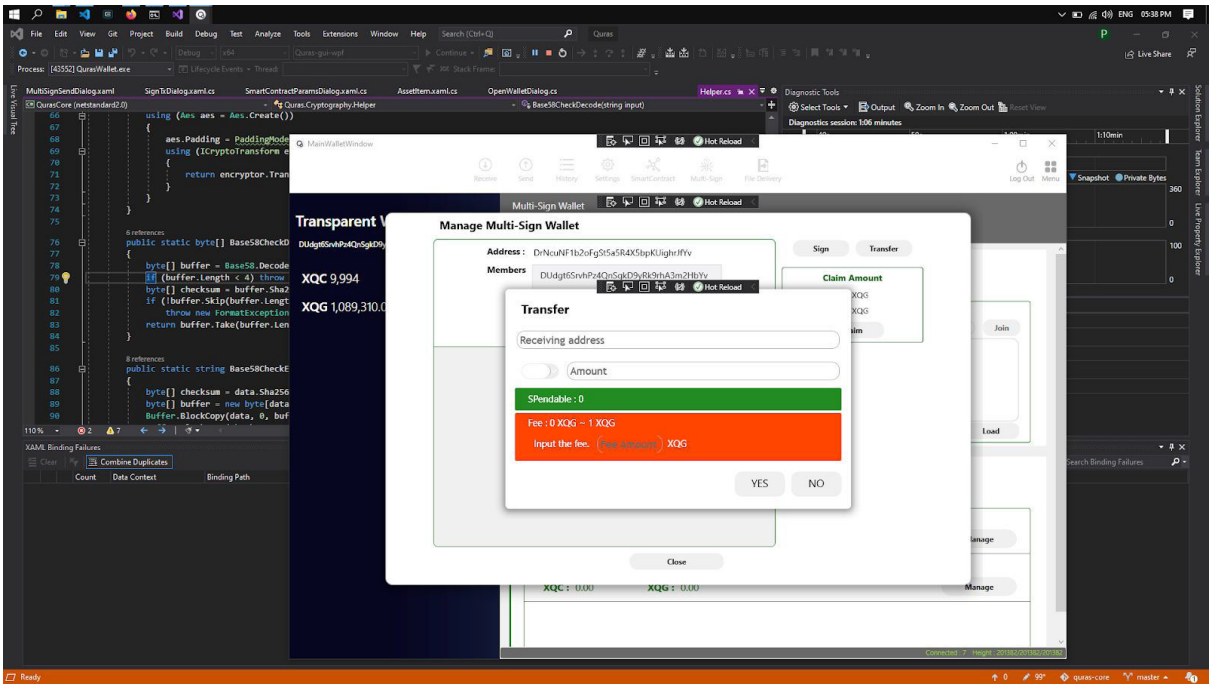


```
using System.Linq;
using System.Numerics;

namespace Quras_gui_wpf.Global
{
    internal class AssetDescriptor
    {
        public UIntBase AssetId;
        public string AssetName;
        public byte Precision;

        public AssetDescriptor(UInt160 asset_id)
        {
            byte[] script;
            using (ScriptBuilder sb = new ScriptBuilder())
            {
                sb.EmitAppCall(asset_id, "decimals");
                sb.EmitAppCall(asset_id, "name");
                script = sb.ToArray();
            }
            ApplicationEngine engine = ApplicationEngine.Run(script);
            if (engine.State.HasFlag(VMState.FAULT)) throw new ArgumentException();
            this.AssetId = asset_id;
            this.AssetName = engine.EvaluationStack.Pop().GetString();
            this.Precision = (byte)engine.EvaluationStack.Pop().GetBigInteger();
        }

        public AssetDescriptor(AssetState state)
        {
            this.AssetId = state.AssetId;
            this.AssetName = state.GetName();
            this.Precision = state.Precision;
        }

        public Global.BigDecimal GetAvailable()
        {
            if (AssetId is UInt256 asset_id)
            {
                return new Global.BigDecimal(Constant.CurrentWallet.GetAvailable(asset_id).GetData(), 8);
            }
            else
            {
                byte[] script;
                using (ScriptBuilder sb = new ScriptBuilder())
```

3) Transfer crashes for multisign when wallet is empty

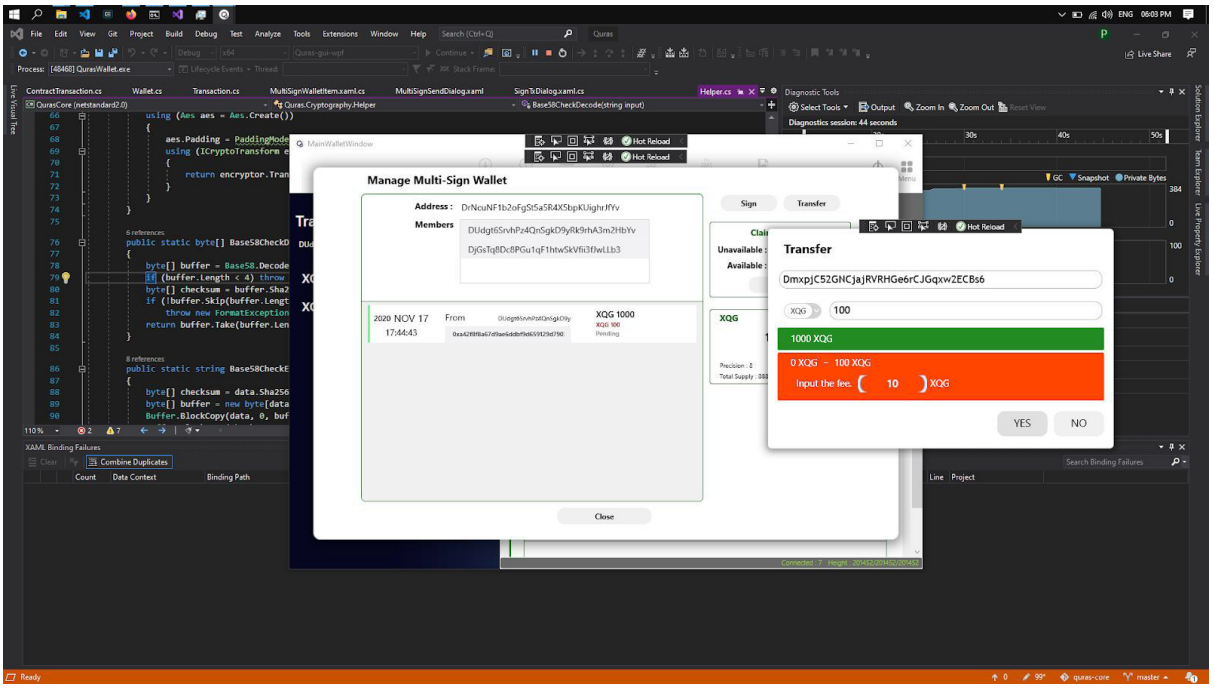
Medium





Reason :
Invalid
UInt256 assetId = ((AssetTypeItem)((ComboBoxItem)cmbAssetType.SelectedItem).Tag).AssetID;

4) **Transfer between multisign wallets crashes the application** Medium



```
if (displayed_dlg != null)
    displayed_dlg.RefreshAssets(Assets);
}

1 reference
private void BtnManage_Click(object sender, RoutedEventArgs e)
{
    using (MultiSignWalletManageDialog dlg = new MultiSignWalletManageDialog { Owner = System.Windows.Application.Current.MainWindow })
    {
        dlg.RegisterTx = RegisterTx;
        dlg.Assets = Assets;
        dlg.RefreshHistoryItems(TxHistoryItems);
        displayed_dlg = dlg;
        _ = dlg.ShowDialog(); // txn object has no system fee
        displayed_dlg = null;
    }
}

2 references
public void AddTransaction(TransactionInfo info)
{
    bool isAdd = true;
    foreach (HistoryItem item in TxHistoryItems)
    {
```

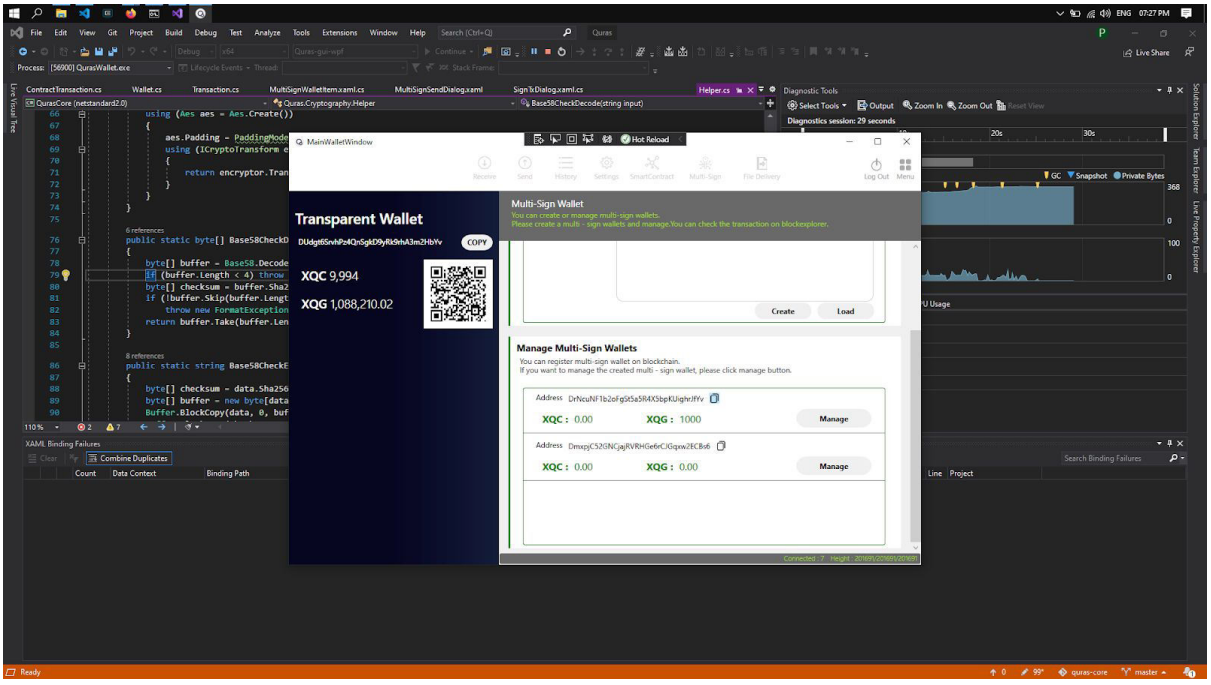
5) **Serve updates over https** Note

Currently the updates are being served over an http server, we highly recommend to serve updates over https. This is only as a note but is highly recommended to be followed.

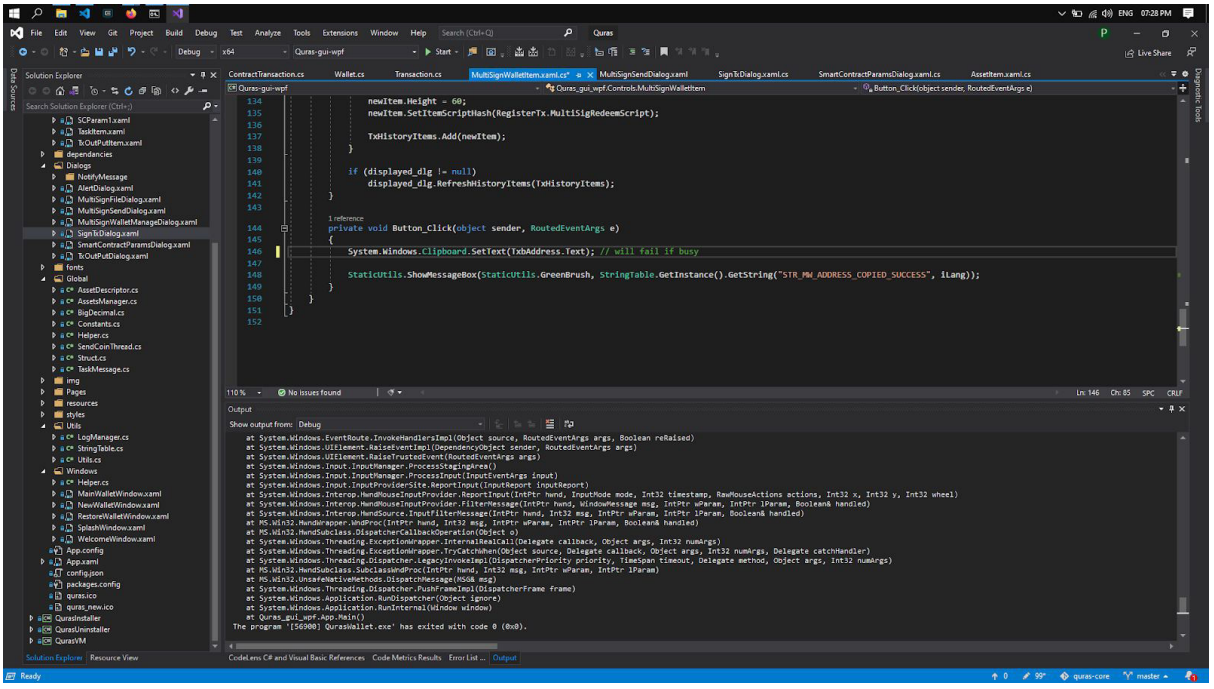
```
namespace Quras_gui_wpf.Global
{
    99+ references
    public static class Constant
    {
        public static string PEER_STATE_PATH = "peers.dat";
        public static LocalNode LocalNode;
        public static UserWallet CurrentWallet;
        public static bool bSnarksParamLoaded = false;
        public static NotifyMessageManager NotifyMessageMgr;
        public static string UpdateUrl = "http://blockapi.quras.io/update/update.xml";
        public static List<TaskMessage> TaskMessages = new List<TaskMessage>();
        public static bool isLoadedVK = false;
        public static bool isLoadedPK = false;
        public static XDocument xdoc = null;

        37 references
        public static LANG GetLang()
        {
            LANG iLang = LANG.EN;

            if (Settings.Default.Language == "EN")
            {
                iLang = LANG.EN;
            }
            else if (Settings.Default.Language == "JP")
            {
                iLang = LANG.JP;
            }

            return iLang;
        }
```

6) **Denial of service** High

Application will crash if another application is accessing clipboard (*To replicate it, try clicking on the following copy button continuously and fast.)

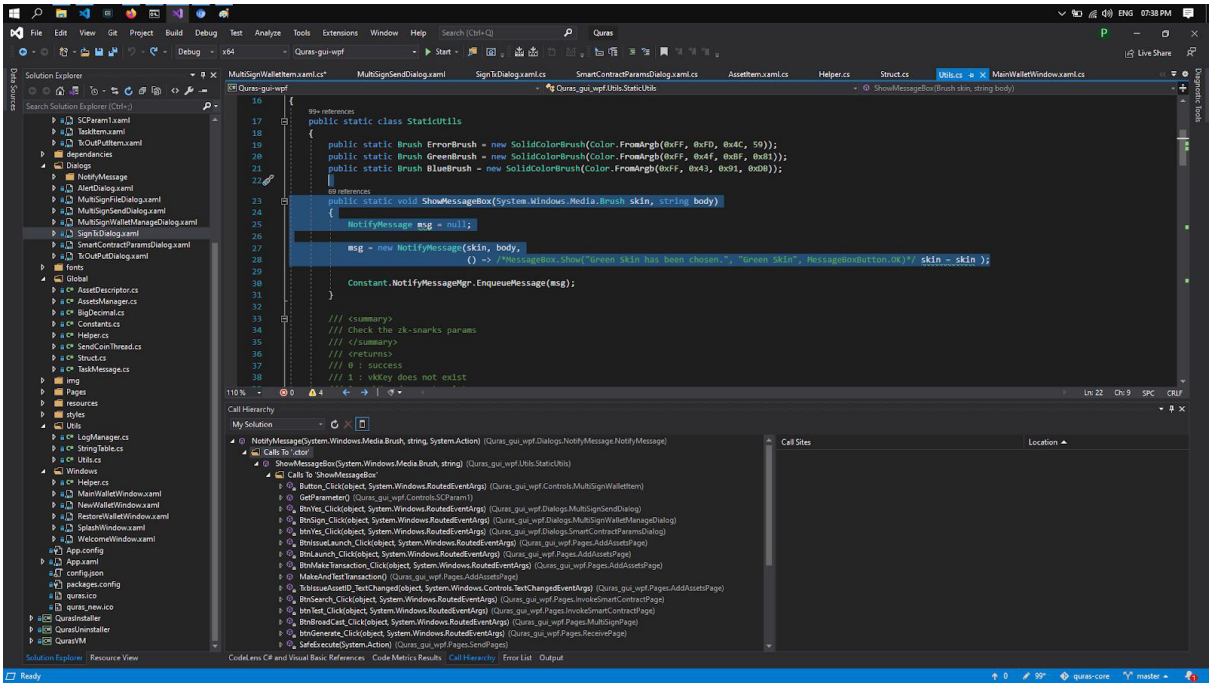


Any other application can cause denial of service to the Quras exe simply by flooding the OS clipboard when the application is in use and the user tries to copy an address.



7) Unnecessary assignment to variable

NOTE



In “notify” message, “skin” variable is assigned to self. That might not be as intended.

Ceremony

https://github.com/quras-official/quras-ceremony
Conducted in November 2020

Concise security analysis

1. Weak Random number generator	Note
2. Base58 encoding is wrong	Medium
3. Incorrect Bloom Filter check	Low
4. Incorrect ECC check	Medium
5. Use of HTTP mode	Medium
6. Use of ECB mode	Low
7. Use of RijndaelManaged	Medium
8. SMTP credential leak	Critical
9. SQL Injection	Critical
10. Same Variable Assignment	Note

Note	2
Low	2
Medium	4
Critical	2

Priority : Note
Issue : Weak Random number generator is used for generating big integers. This is a minor note level issue only as it looks like this function is not being used anywhere.

CeremonyClient\IO\Helper.cs
https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/IO/Helper.cs#L109

```
internal static BigInteger NextBigInteger(this Random rand, int sizeInBits)
{
    if (sizeInBits < 0)
        throw new ArgumentException("sizeInBits must be non-negative");
    if (sizeInBits == 0)
        return 0;
    byte[] b = new byte[sizeInBits / 8 + 1];
    rand.NextBytes(b);
    if (sizeInBits % 8 == 0)
        b[b.Length - 1] = 0;
    else
        b[b.Length - 1] &= (byte)((1 << sizeInBits % 8) - 1);
    return new BigInteger(b);
}
```

Priority : Medium
Issue : Base58 encoding is wrong.

CeremonyClient\Cryptography\Base58.cs
https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/Cryptography/Base58.cs

```
private void LocalNode_InventoryReceived(object sender, IInventory inventory)
{
    ConsensusPayload payload = inventory as ConsensusPayload;
    if (payload != null)
    {
        lock (context)
        {
            if (payload.ValidatorIndex == context.MyIndex) return;
            if (payload.Version != ConsensusContext.Version || payload.PrevHash != context.PrevHash || payload.BlockIndex != context.BlockIndex)
                return;
            if (payload.ValidatorIndex >= context.Validators.Length) return;
            ConsensusMessage message;
            try
            {
                message = ConsensusMessage.DeserializeFrom(payload.Data);
            }
            catch (FormatException)
            {
                return;
            }
```

http://lenschulwitz.com/base58
https://rextester.com/ZMS14027
Please use the link above to check the values of base58 and in the second link we have provided the reason for it to be still wrong (check the key and value outputs).

Priority : Low

Issue : Bloom filters should check for valid m & k values before generating seed, this function is not being used hence has been kept in not only specification.

CeremonyClient/Cryptography/BloomFilter.cs

https://github.com/quras-official/quras-ceremony/blob/master/CeremonyServer/Cryptography/BloomFilter.cs

```
public BloomFilter(int m, int k, uint nTweak, byte[] elements = null)
{
    this.seeds = Enumerable.Range(0, k).Select(p => (uint)p * 0xFBA4C795 + nTweak).ToArray();
    this.bits = elements == null ? new BitArray(m) : new BitArray(elements);
    this.bits.Length = m;
    this.Tweak = nTweak;
}
```

Priority : Medium

Issue : Invalid ECC comparisons and generation, here a point is returned for an invalid curve also , the comparison is also faulty. The validity of the curve is not taken into consideration.

CeremonyClient\Cryptography\ECC\ECPoint.cs

https://github.com/quras-official/quras-ceremony/blob/master/CeremonyServer/Cryptography/ECC/ECPoint.cs

```
7 references
internal ECPoint(ECFieldElement x, ECFieldElement y, ECCurve curve)
{
    if ((x != null && y == null) || (x == null && y != null))
        throw new ArgumentException("Exactly one of the field elements is null");
    this.X = x;
    this.Y = y;
    this.Curve = curve;
}

/// <summary>
/// Compare with another object
/// </summary>
/// <param name="other">Another object</param>
/// <returns>Return the result of the comparison</returns>
62 references
public int CompareTo(ECPoint other)
{
    if (ReferenceEquals(this, other)) return 0;
    int result = X.CompareTo(other.X);
    if (result != 0) return result;
    return Y.CompareTo(other.Y);
}
```

Priority : Medium
Issue : HTTP is not suggested at all for any communication and should be shifted to use HTTPS mode to prevent any man in the middle attacks.

CeremonyClient\Network\RpcClient.cs

https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/Network/RpcClient.cs

```
5 references
public string SendRequest(string queryString, string encodeType = "UTF-8")
{
    Console.WriteLine(queryString);
    Console.WriteLine(ServerUrl);
    Task<string> task = Task<string>.Factory.StartNew(() =>
    {
        WebRequest request = WebRequest.Create(ServerUrl);
        request.ContentType = "application/json";
        request.Method = "POST";

        byte[] buffer = Encoding.GetEncoding(encodeType).GetBytes(queryString);
        string result = System.Convert.ToBase64String(buffer);
        Stream reqstr = request.GetRequestStream();
        Console.WriteLine(buffer.ToString());

        reqstr.Write(buffer, 0, buffer.Length);
        reqstr.Close();
```

Priority : Low
Issue : The ECB mode is prone to various crypto attacks. Use a stronger mode such as CBC instead.

https://github.com/quras-official/quras-ceremony/blob/master/CeremonyServer/Cryptography/Helper.cs#L34

```
internal static byte[] AES256Encrypt(this byte[] block, byte[] key)
{
    using (Aes aes = Aes.Create())
    {
        aes.Key = key;
        aes.Mode = CipherMode.ECB;
        aes.Padding = PaddingMode.None;
        using (ICryptoTransform encryptor = aes.CreateEncryptor())
        {
            return encryptor.TransformFinalBlock(block, 0, block.Length);
        }
    }
}
```

Priority : Medium

Issue : RijndaelManaged is being used to save Wallet and is not secure for production systems. It uses a Microsoft proprietary extended PBKDF1 implementation of PasswordDeriveBytes instead of PBKDF2. This implementation is not secure for any bytes over 20 bytes long as there may even be repeated bytes in the output.
Also any output (with size over 20 bytes) won't be reproducible in any other framework.
We highly suggest to move away from RijndaelManaged during wallet operations.

https://github.com/quras-official/quras-ceremony/blob/b4e1ba32d8baf09c30687cb35f3c07b91140faf0/CeremonyServer/Wallets/Wallet.cs#L40

```
public void SaveWallet(string walletName, string walletPassword)
{
    //generate random salt
    byte[] salt = GenerateRandomSalt();

    //create output file name
    FileStream fsCrypt = new FileStream(walletName + ".aes", FileMode.Create);

    //convert password string to byte arrray
    byte[] passwordBytes = System.Text.Encoding.UTF8.GetBytes(walletPassword);

    //Set Rijndael symmetric encryption algorithm
    RijndaelManaged AES = new RijndaelManaged();
    AES.KeySize = 256;
    AES.BlockSize = 128;
    AES.Padding = PaddingMode.PKCS7;

    //"What it does is repeatedly hash the user password along with the salt." High iteration counts.
    var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000);
    AES.Key = key.GetBytes(AES.KeySize / 8);
    AES.IV = key.GetBytes(AES.BlockSize / 8);

    AES.Mode = CipherMode.CFB;

    fsCrypt.Write(salt, 0, salt.Length);

    CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateEncryptor(), CryptoStreamMode.Write);

    try
    {
        cs.Write(walletKey.PrivateKey, 0, walletKey.PrivateKey.Length);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: " + ex.Message);
```

Priority : Critical
Issue : The code includes the password to SMTP server in the settings file of the server. This can lead to trojan attacks where attackers can ask participants to install software that might compromise the Ceremony. The critical information like this should be part of session information on the server and passed only as a variable.

We also suggest removing the current commit for password and creating a new repository. Otherwise a proof that some malicious activity has already not taken place must be proved.

```
namespace CeremonyServer
{
    7 references
    internal sealed partial class Settings
    {
        public static Settings instance;

        private const string EmailAddress = "tech@quras.io";
        private const string EmailPassword = "0tJhaxZnyURus8tl8f";

        3 references
        public static Settings Default
        {
            get
            {
                if (instance == null)
                    instance = new Settings();

                return instance;
            }
        }
```

Priority : Critical
Issue : SQL injection in the server can lead to compromisation of the Ceremony. The server takes input from the user without validating the inputs or passing them in a secure way. This can lead to server information takeover and a single person will be able to modify the keys to control the generation of QURAS coins.

```
public User SignInUser(JObject param)
{
    User ret = new User();
    MySqlDataReader rdr = null;

    try
    {
        command.CommandText = "SELECT * From tbl_user " +
            "WHERE email='" + param["email"].AsString() + "' " +
            "AND password=MD5('" + param["password"].AsString() + "')";

        rdr = command.ExecuteReader();

        if (!rdr.Read())
        {
            throw new Exception("No Selected User");
        }

        ret.Id = rdr.GetInt32("id");
        ret.Name = rdr.GetString("name");
```

The malicious user can execute another process even before the server starts the ceremony.

https://github.com/quras-official/quras-ceremony/blob/master/CeremonyServer/IO/MySQL/CeremonySQL.cs

Priority: Note

Same value is assigned to itself and has no effect on the code as intended.

https://github.com/quras-official/quras-ceremony/blob/b4e1ba32d8baf09c30687cb35f3c07b91140faf0/CeremonyClient/Utils/StaticUtils.cs

```
5 references
public static void ShowMessageBox(System.Windows.Media.Brush skin, string body)
{
    NotifyMessage msg = null;

    msg = new NotifyMessage(skin, body,
                            () => /*MessageBox.Show("Green Skin has been chosen.", "Green Skin", MessageBoxButton.OK)*/ skin = skin);

    NotifyMessageMgr.EnqueueMessage(msg);
}
```