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)

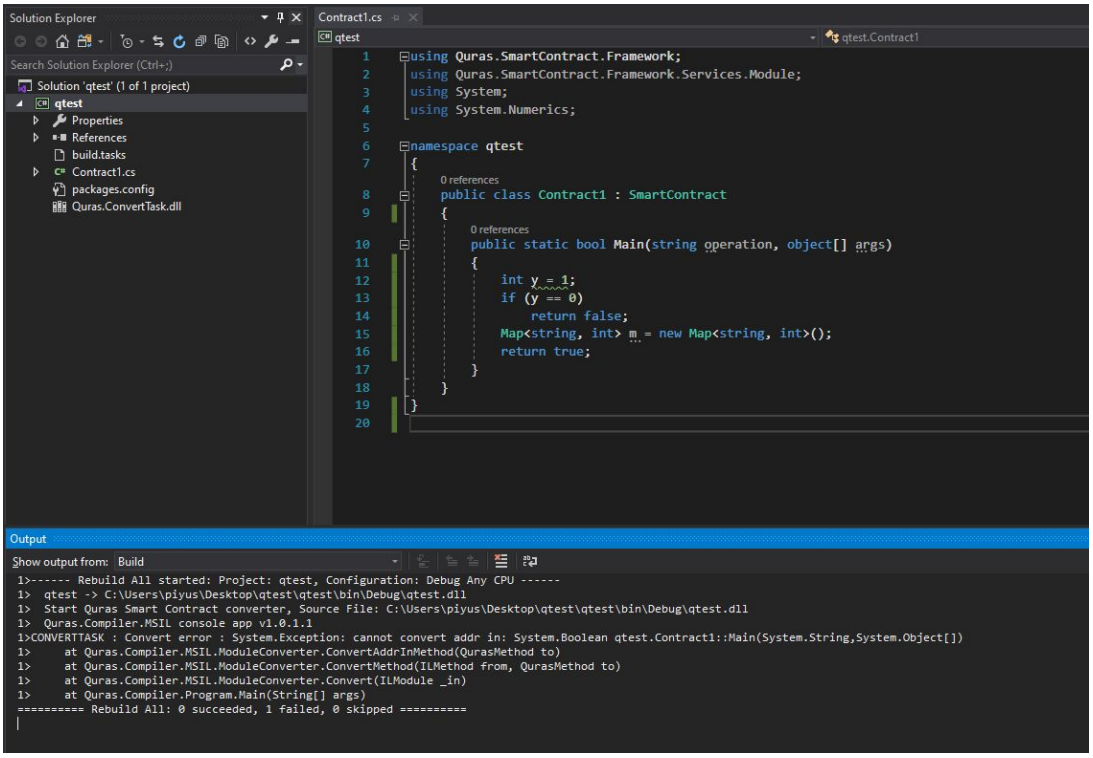
_Convert1by1(Quras.VM.OpCode.INVERT, src, to);
_Insert1(Quras.VM.OpCode.EQUAL, "", to);

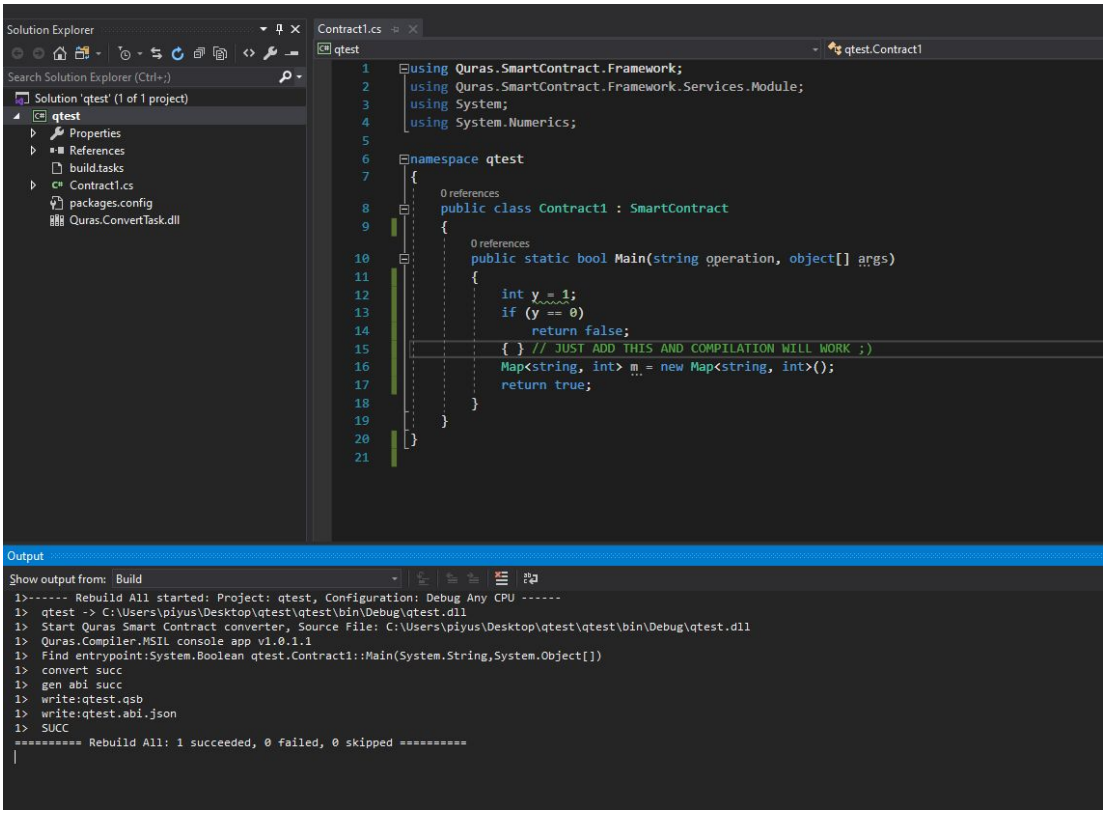
Should be changed to :

_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
```

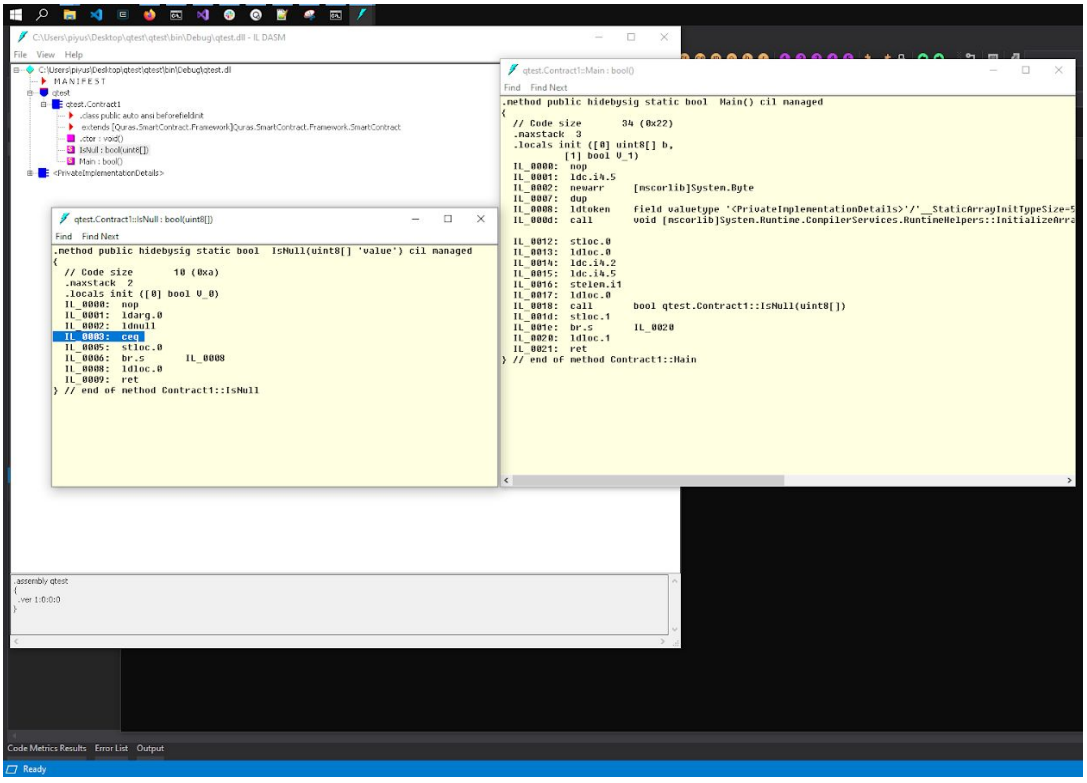
Here in line 29 SETITEM fails as it is allowed for arrays or maps (not bytearray).

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;
}
```