# Ceremony

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

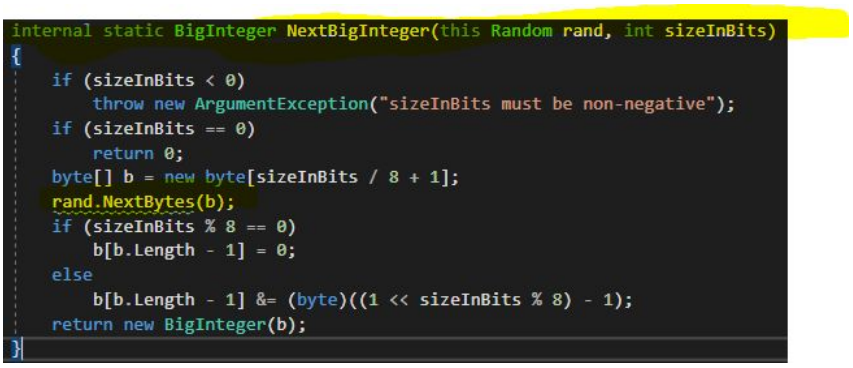
## Concise security analysis

| | | |
|---|---|---|
| 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.

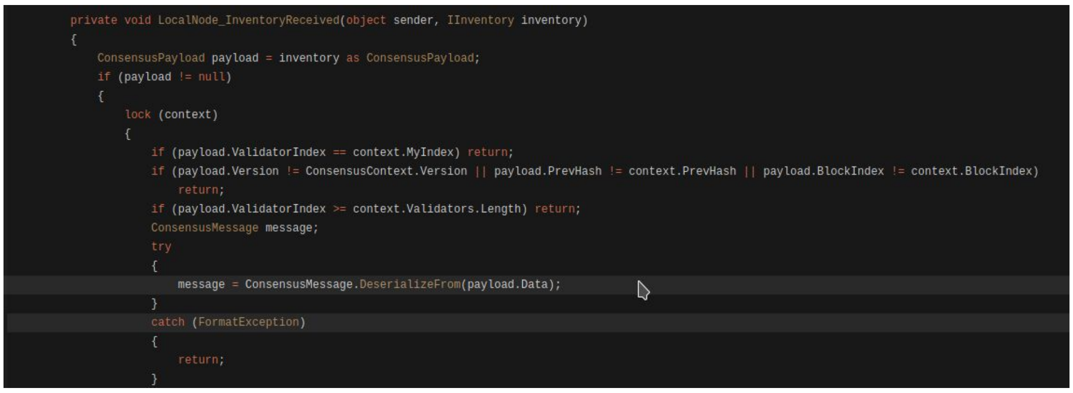
*CeremonyClient\IO\Helper.cs*
*[https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/IO/Helper.cs#L109](https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/IO/Helper.cs#L109)*



```
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](https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/Cryptography/Base58.cs)



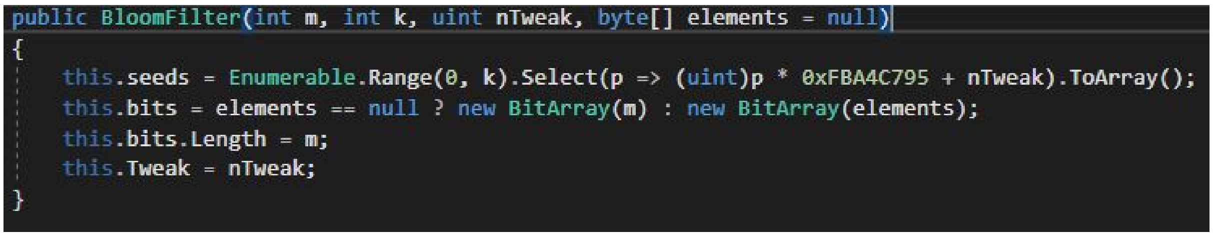
```
private void LocalNode_InventoryReceived(object sender, IInventory inventory)
{
    ConsensusPayload payload = inventory as ConsensusPayload;
    if (payload != null)
    {
        lock (context)
        {
            if (payload.ValidatorIndex == context.MyIndex) return;
            if (payload.Version != 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](http://lenschulwitz.com/base58)
[https://rextester.com/ZMS14027](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](https://github.com/quras-official/quras-ceremony/blob/master/CeremonyServer/Cryptography/BloomFilter.cs)

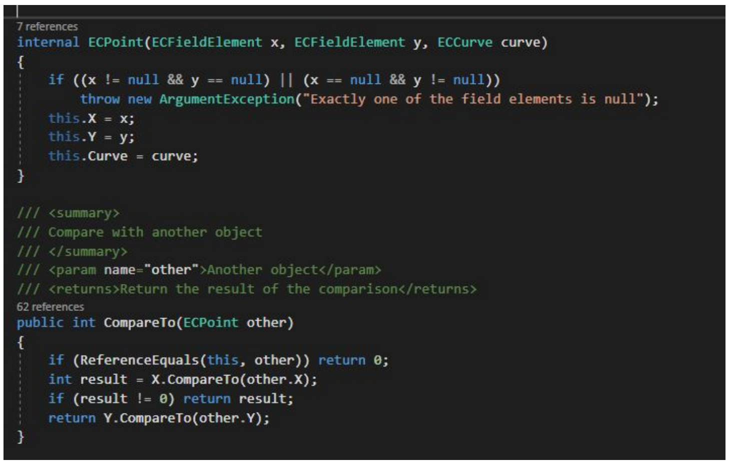


```
public BloomFilter(int m, int k, uint nTweak, byte[] elements = null)
{
    this.seeds = Enumerable.Range(0, k).Select(p => (uint)p * 0xFBA4C795 + nTweak).ToArray();
    this.bits = elements == null ? new BitArray(m) : new BitArray(elements);
    this.bits.Length = m;
    this.Tweak = nTweak;
}
```

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

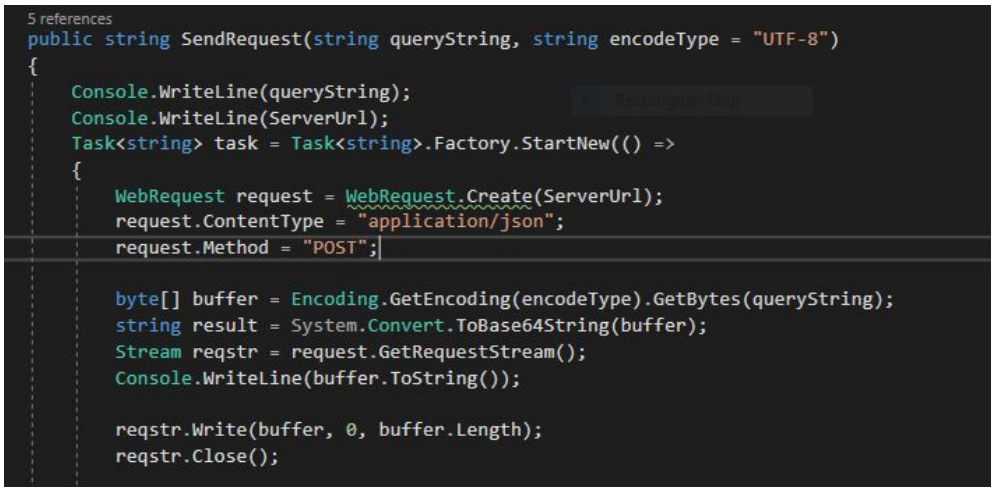


```
7 references
internal ECPoint(ECFieldElement x, ECFieldElement y, ECCurve curve)
{
    if ((x != null && y == null) || (x == null && y != null))
        throw new ArgumentException("Exactly one of the field elements is null");
    this.X = x;
    this.Y = y;
    this.Curve = curve;
}

/// <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*

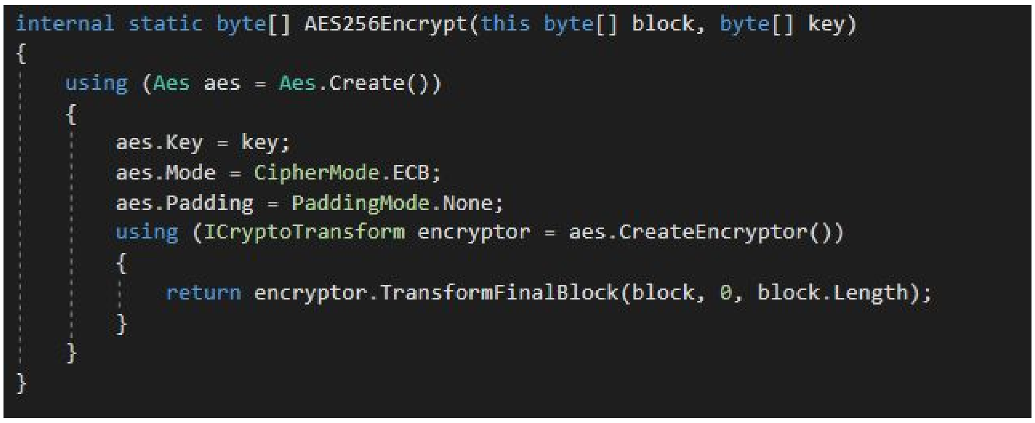
https://github.com/quras-official/quras-ceremony/blob/master/CeremonyClientFinal/Network/RpcClient.cs



```
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**

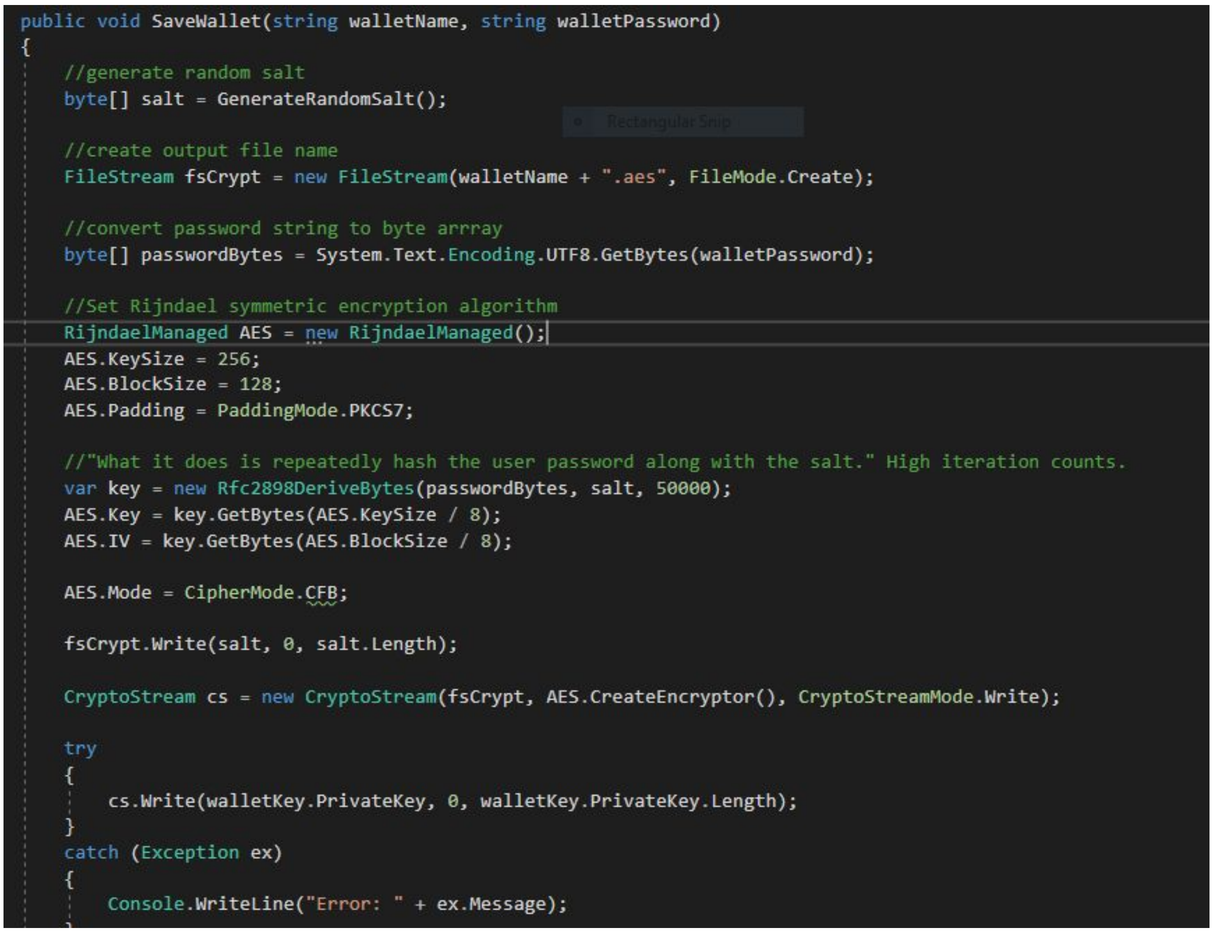


```
internal static byte[] AES256Encrypt(this byte[] block, byte[] key)
{
    using (Aes aes = Aes.Create())
    {
        aes.Key = key;
        aes.Mode = CipherMode.ECB;
        aes.Padding = PaddingMode.None;
        using (ICryptoTransform encryptor = aes.CreateEncryptor())
        {
          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

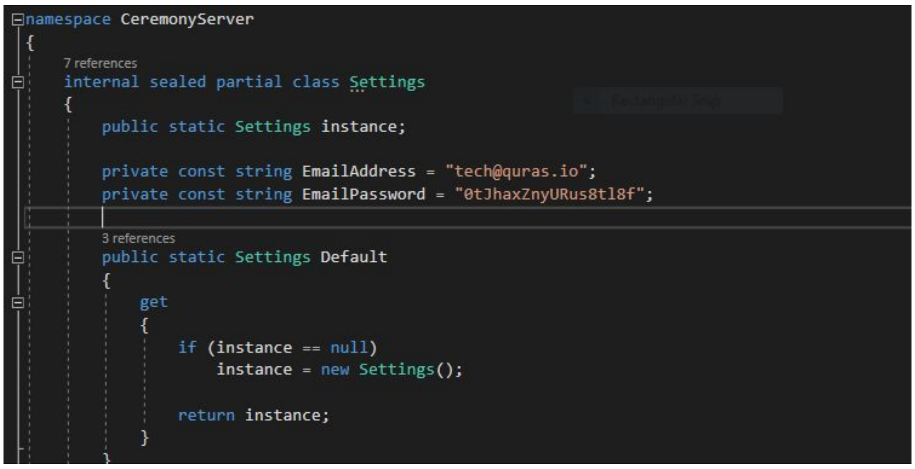
[https://github.com/quras-official/quras-ceremony/blob/b4e1ba32d8baf09c30687cb35f3c07b91140faf0/CeremonyServer/Wallets/Wallet.cs#L40](https://github.com/quras-official/quras-ceremony/blob/b4e1ba32d8baf09c30687cb35f3c07b91140faf0/CeremonyServer/Wallets/Wallet.cs#L40)



```
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);
```

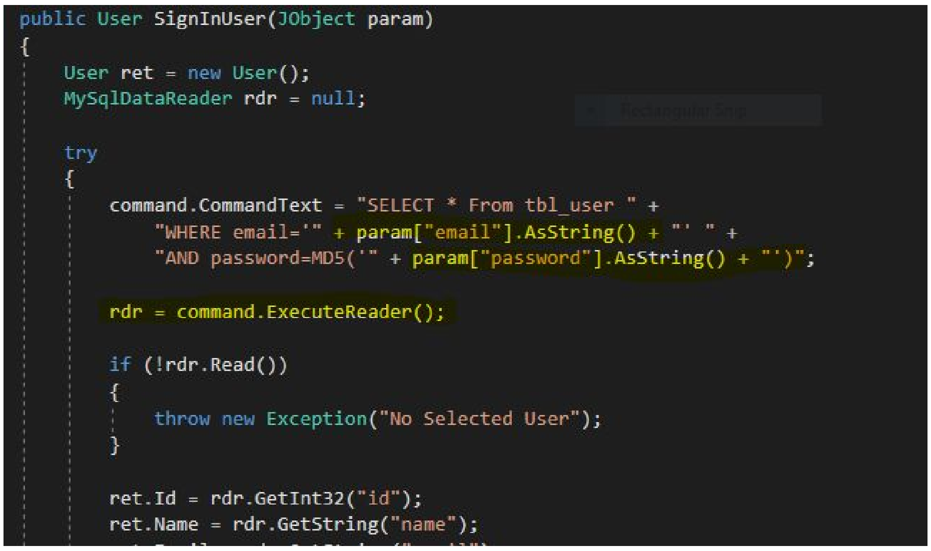
**Priority : Critical**
Issue : The code includes the password to SMTP server in the settings file of the server. This can lead to trojan attacks where attackers can ask participants to install software that might compromise the Ceremony. The critical information like this should be part of session information 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");
```

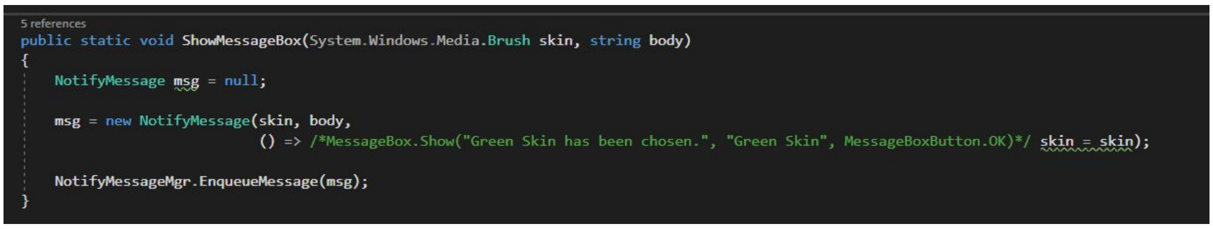
The malicious user can execute another process even before the server starts the ceremony.

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);
}
```