

# VULNERABILITY RESEARCH REPORT

## Uncontrolled Resource Consumption via Malformed .sla Project File

Scribus Desktop Publishing Software — Version 1.6.5

Reported by: Dr. Mohammadreza Ashouri | ByteScan.net | audit@bytescan.net

### 1. Vulnerability Summary

<b>Title</b>	Uncontrolled Resource Consumption (CPU/Memory) via Malformed .sla File
<b>Software</b>	Scribus Desktop Publishing Software
<b>Affected Version</b>	1.6.5
<b>Platform</b>	macOS 12.3 Monterey (arm64e, x86_64 via Rosetta 2)
<b>File Type</b>	.sla (Scribus XML project file)
<b>CWE</b>	CWE-400: Uncontrolled Resource Consumption / CWE-20: Improper Input Validation
<b>Severity</b>	High
<b>CVSS v3.1 Score</b>	7.1 (AV:L/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:H)
<b>Researcher</b>	Dr. Mohammadreza Ashouri — ByteScan Security Research
<b>Contact</b>	audit@bytescan.net
<b>Report Date</b>	2026-03-25
<b>Disclosure Type</b>	Coordinated Responsible Disclosure

### 2. Affected Component

Scribus is a free and open-source desktop publishing application. It uses an XML-based project file format with the extension .sla. When a project file is opened, Scribus parses all XML attributes including numeric geometry fields such as page dimensions, object positions, and frame sizes, and it passes these values directly into its layout engine without bounds validation and the problem lies here.

<b>Application</b>	Scribus
<b>Version</b>	1.6.5
<b>Bundle ID</b>	net.scribus

Architecture	x86_64
Qt Version	5.15.12
Build SDK	macOS 13.1 Ventura
Min Deployment Target	macOS 12.0 Monterey
Compiler/Linker	Apple Clang, Apple Id (tool ID 3, Xcode 14.2)
Debug Symbols	Stripped (release build)

### 3. Vulnerability Description

#### 3.1 Root Cause

Scribus parses numeric XML attributes from .sla project files without enforcing upper or lower bounds on geometric values. Attributes including PAGEWIDTH, PAGEHEIGHT, BORDERLEFT, PAGEXPOS, PAGEYPOS, WIDTH, HEIGHT, YPOS, and gYpos are read directly and passed into Qt geometry structures (QRect, QRegion) without any sanitization. When abnormally large values are provided — for example WIDTH="1444444444400" or YPOS="9444445.9999985" — the text frame layout engine enters an infinite loop during region containment checking, consuming 99% CPU indefinitely and causing a system-wide memory pressure cascade.

#### 3.2 Vulnerable Code Path

The following call stack was captured in the macOS during our fuzzing test cpu\_resource.diag report (incident BE36427C-EBDC-4436-8528-F7604B7E09F5):

```

ScribusMainWindow::loadDoc(QString const&)          [Scribus + 3573169]
  PageItem_TextFrame::layout()                    [Scribus + 1959695]
    regionContainsRect(QRegion const&, QRect)      [Scribus + 4614563]  <-- LOOP
      QRect::contains(QRect const&, bool) const   [QtCore + 346712]
        QRegion::boundingRect() const
          QRegion::rectCount() const

```

All 32 CPU samples (100%) taken over 82.93 seconds were inside regionContainsRect(). The function never returned, that means an infinite loop caused by geometry values exceeding the range that the Qt region containment algorithm is designed to handle.

### 3.3 Trigger Mechanism

The vulnerability is triggered simply by opening a crafted .sla file in Scribus. No user interaction beyond the file open action is required. The malicious file can be delivered via email attachment, downloaded file, shared network path, or any other standard file distribution mechanism. The relevant malformed fields in the proof-of-concept file are:

```
PAGEOBJECT YPOS="9444445.9999985582247"  
  
    WIDTH="1444444444400"  
  
    HEIGHT="144444444400"  
  
    gYpos="924242424242424245.9999985582247"  
  
PAGE    PAGEXPOS="144444400"  
  
        PAGEWIDTH="64444444412"  
  
MASTERPAGE PAGEXPOS="100098989898989898989890"  
  
        PAGEHEIGHT="799868689689689689689689689689686986986896986962"
```

## 4. Impact

### 4.1 Application-Level Impact

The Scribus process consumed 99% CPU for the entire duration of the monitoring window (90 seconds), rendering the application completely unresponsive. The macOS kernel generated a `cpu_resource.diag` report confirming the violation:

```
CPU used:      90 seconds  
  
CPU duration: 91 seconds  
  
CPU average:  99%  
  
Limit:        50% CPU over 180 seconds  
  
Action:       none (application not terminated by OS)
```

## 4.2 System-Level Impact — Jetsam Memory Cascade

The memory exhaustion caused by Scribus processing the oversized geometry values triggered the macOS jetsam memory manager to begin force-killing idle system processes. The following processes were terminated by the kernel between 17:58:06 and 17:59:46 UTC on 2026-03-25:

Process	Exit Reason
PodcastContentService	JETSAM_REASON_MEMORY_IDLE_EXIT
AMPartworkAgent	JETSAM_REASON_MEMORY_IDLE_EXIT
sysextd	JETSAM_REASON_MEMORY_IDLE_EXIT
secinitd (x2)	JETSAM_REASON_MEMORY_IDLE_EXIT
containermanagerd (x3)	JETSAM_REASON_MEMORY_IDLE_EXIT
ReportCrash	JETSAM_REASON_MEMORY_IDLE_EXIT
photolibraryd	JETSAM_REASON_MEMORY_IDLE_EXIT
softwareupdated (x2)	JETSAM_REASON_MEMORY_IDLE_EXIT
TrustedPeersHelper	JETSAM_REASON_MEMORY_IDLE_EXIT
CloudKeychainProxy	JETSAM_REASON_MEMORY_IDLE_EXIT
XprotectService (x2)	JETSAM_REASON_MEMORY_IDLE_EXIT
HandBrakeXPCService	JETSAM_REASON_MEMORY_IDLE_EXIT
biometrickitd	JETSAM_REASON_MEMORY_IDLE_EXIT
40+ additional processes	JETSAM_REASON_MEMORY_IDLE_EXIT

This constitutes a Denial of Service at the operating system level. A single malicious file opened by a regular user triggers a cascade that disrupts unrelated system services, security daemons (biometrickitd, secinitd, TrustedPeersHelper), and crash reporting infrastructure (ReportCrash).

## 4.3 Secondary Finding — QPixmap Dimension Mismatch

The Qt logging system also recorded the following error twice during document load, caused by the oversized geometry values being passed into Qt's pixmap subsystem:

```
Scribus: (QtCore) QPixmap::setMask() mask size differs from pixmap size
```

This indicates that the malformed values propagate through multiple subsystems beyond the text frame layout engine.

## 5. Proof of Concept

### 5.1 PoC File

The proof-of-concept is a valid Scribus .sla XML file (well-formed XML, accepted by Scribus's parser) with the following mutated numeric attributes in the PAGEOBJECT element:

```
<?xml version="1.0" encoding="UTF-8"?>
<SCRIBUSUTF8NEW Version="1.6.5">
  <DOCUMENT PAGEWIDTH="612" PAGEHEIGHT="792" ...>
    <PAGEOBJECT
      XPOS="295.905877908881"
      YPOS="94444445.9999985582247"
      WIDTH="14444444444400"
      HEIGHT="1444444444400"
      gYpos="924242424242424245.9999985582247"
      ...>
      <StoryText>
        <ITEXT CH="HelloWorld..." />
      </StoryText>
    </PAGEOBJECT>
  </DOCUMENT>
</SCRIBUSUTF8NEW>
```

### 5.2 Reproduction Steps

1. Install Scribus 1.6.5 on macOS 12.x
2. Open the attached PoC .sla file via File > Open

3. Observe: application immediately hangs, CPU rises to 99%

4. After ~3 minutes, macOS begins killing idle system processes

5. Crash report generated at:

`/Library/Logs/DiagnosticReports/Scribus_<timestamp>_<hostname>.cpu_resource.diag`

### 5.3 Diagnostic Report Reference

Incident ID	BE36427C-EBDC-4436-8528-F7604B7E09F5
Report Type	cpu_resource.diag
Timestamp	2026-03-25 17:53:54 UTC
Duration Sampled	82.93 seconds
CPU Usage	90s over 91s (99% average)
Memory Footprint	317.30 MB at time of report
Samples in loop	32/32 (100%) inside regionContainsRect()

## 6. CVSS v3.1 Scoring

Attack Vector (AV)	Local (L) — requires opening a file
Attack Complexity (AC)	Low (L) — no special conditions
Privileges Required (PR)	None (N) — no privileges needed
User Interaction (UI)	Required (R) — user must open the file
Scope (S)	Changed (C) — system processes affected beyond Scribus
Confidentiality (C)	None (N)
Integrity (I)	None (N)
Availability (A)	High (H) — system-wide DoS
Base Score	7.1 — HIGH
Vector String	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:C/C:N/I:N/A:H

## 7. Remediation Recommendation

The fix requires input validation on all numeric XML attributes parsed from .sla files before they are passed to Qt geometry functions. Recommended mitigations:

1. Bounds checking on geometry attributes meaning putting an enforce maximum values on PAGEWIDTH, PAGEHEIGHT, WIDTH, HEIGHT, XPOS, YPOS, PAGEXPOS, PAGEYPOS, gXpos, gYpos, BORDERLEFT, BORDERRIGHT, BORDERTOP, BORDERBOTTOM. Reasonable maximums should reflect the application's maximum supported canvas size.

2. Integer overflow protection: values that exceed int32 or double precision range should be rejected with an error dialog rather than passed to the layout engine.

3. Loop guard in regionContainsRect() : add an iteration counter with a maximum bound to prevent infinite looping regardless of input values.

4. Fuzz testing : the .sla parser should be included in a fuzzing campaign targeting all numeric XML attributes to identify similar issues in other fields.

**Suggested fix location in Scribus source:**

scribus/scribusdoc.cpp - document loading / attribute parsing

scribus/pageitem\_textframe.cpp - PageItem\_TextFrame::layout()

scribus/util\_render.cpp - regionContainsRect()

## 8. Researcher Information

<b>Researcher</b>	Dr. Mohammadreza Ashouri
<b>Organization</b>	ByteScan Security Research
<b>Website</b>	<a href="https://bytescan.net">https://bytescan.net</a>
<b>Email</b>	<a href="mailto:audit@bytescan.net">audit@bytescan.net</a>

**ByteScan is a cybersecurity lab specializing in vulnerability research, smart contract security audits, penetration testing, and responsible disclosure. This research was conducted as part of ongoing desktop application security research targeting project file parsers.**

ByteScan.net | audit@bytescan.net | March 2026