Solution to ImageMagick "not authorized" PDF Error

ImageMagick "not authorized" PDF errors

Here's the scenario: I have scanned a multi-page document using Xsane. I named the first one `scan-0001.jpg` and Xsane automatically incremented the number for each new file. Now I want to convert that collection of JPEG images into one PDF file. This used to work, but now I just get an error:

```
$ ls
scan-0001.jpg   scan-0002.jpg   scan-0003.jpg   scan-0004.jpg
$ convert scan*.jpg scan.pdf
convert-im6.q16: not authorized `scan.pdf' @ error/constitute.c/WriteImage/1037.
```

What has happened?

ImageMagick has supported a security policy for some time. The ImageMagick project itself, and the Linux distributions in which it's included, have made the policy a little more strict. Now it refuses to write PDF and PostScript files on my system. The error message is true, but it isn't helpful. I don't care precisely *where* within the code that it was denied, I just want to know *how* to turn this on! ***However, that might be dangerous.***

There are severe risks associated with PostScript data for both servers and desktops. Read on to see what the risks are. There are workarounds, so *maybe* you will want to enable PDF and PostScript processing by ImageMagick. But make sure you understand the risks before continuing.

Some Data Can Be Dangerous

Ghostscript project

PostScript is actually a programming language. You know that you can run a tool like `gv` or `ghostview` to view PostScript files, or CUPS or another print service to send something to a printer. But you may not realize that those tools use the Ghostscript libraries, and **Ghostscript runs the PostScript program in the input file.** The usual way of describing this is:

PostScript is a page description language.

Ghostscript is a PostScript interpreter.

PDF or Portable Document Format is based on PostScript. If you use an Adobe tool, it will include its own PostScript interpreter. (And, bring along all the risks of the Adobe suite!)

Open-source PDF tools will rely on Ghostscript.

This is where the danger lies. **When you view a PostScript or PDF file downloaded from the Internet or received from a stranger, you are running a program from an untrusted source.**

Ghostscript includes a simple command-line option, the `-dSAFER` flag. That means "Excecute *most* of the PostScript language, *except do not* execute external programs (like shell escapes) or modify files." The manual page recommends the flag without explaining what is really happening:

```
-dSAFER
       Restricts file operations the job can perform. Strongly recommended
       for spoolers, conversion scripts or other sensitive environments
       where a badly written or malicious PostScript program code must be
       prevented from changing important files.
```

Multiple ways to exploit Ghostscript

Multiple Ghostscript `-dSAFER` sandbox problems

The above sounds good, as far as it goes. However, attackers have found ways to bypass the `-dSAFER` flag. And, more recently, they have found alternative attack vectors that `-dSAFER` doesn't catch. There's a very good argument that this is a language specification flaw in PostScript. **In other words, PostScript defines a language with unfixable security problems.**

Artifex Software, Inc., maintains the Ghostscript project. While they made patches available, they didn't release a new, patched version.

These PostScript problems can open you to attacks in two very different circumstances. First, if you run a server that allows people to upload images for processing. Second, if you use a Linux desktop with the default settings found on many distributions.

Postscript and PDF Risks on Image-Processing Web Servers

Imagine a server that allows users to upload images for processing. Conversion between JPEG and other formats, resizing, or maybe enhancing.

That server probably uses ImageMagick as its back end. ImageMagick doesn't pay attention to file name extensions. It reads the beginning of the file to figure out what format it contains — JPEG, GIF, PNG, whatever. That means that an attacker could rename a hostile PostScript file `whatever.jpg` and upload it, bypassing any intended restrictions for "image data only". The server would give the new input data to ImageMagick, which would run the hostile PostScript program. That could include arbitrary shell commands run with the credentials of the web server process.

ImageMagick uses a policy file to specify which operations are allowed on which data types. **On a server that allows untrusted users to upload data for processing, ImageMagick should have PostScript and PDF disabled.** The file `/etc/ImageMagick-6/policy.xml` should contain:

```
[... lines deleted ...]
<policymap>
  [... lines deleted ...]
  <policy domain="coder" rights="none" pattern="PS" />
  <policy domain="coder" rights="none" pattern="PS2" />
  <policy domain="coder" rights="none" pattern="PS3" />
  <policy domain="coder" rights="none" pattern="EPS" />
  <policy domain="coder" rights="none" pattern="PDF" />
  <policy domain="coder" rights="none" pattern="XPS" />
  [... lines deleted ...]
</policymap>
```

If we instead are talking about your desktop, then you *might* want to enable those data types for ImageMagick.

Re-enabling PostScript and PDF for ImageMagick

ImageMagick Security Policy

ImageMagick's default security policy imposes limits of 256 MiB memory, image dimensions of no more than 8196 pixels high or wide, files can be no larger than 1 GiB, individual tasks can take no more than 120 seconds, and others. See the ImageMagick security policy page for more details. I can increase those limits if I want.

There's an easy way to see the current security policy settings on your system:

```
$ identify -list policy | less
```

If I want to re-enable PostScript and PDF formats for ImageMagick, I could make the following changes in the file `/etc/ImageMagick-6/policy.xml`

```
[... about 70 lines deleted ...]
  <!-- disable ghostscript format types -->
  <!--
  <policy domain="coder" rights="none" pattern="PS" />
  <policy domain="coder" rights="none" pattern="EPI" />
  <policy domain="coder" rights="none" pattern="PDF" />
  <policy domain="coder" rights="none" pattern="XPS" />
  -->
  <policy domain="coder" rights="read|write" pattern="PDF,PS" />
</policymap>
```

I was interested in generating a PDF file from a series of JPEG scans of pages. So, I didn't need to use "read|write" above. I could have allowed only "write" and still accomplished what I want.

Problem Solved!

```
$ convert scan-*jpg scan.pdf
$ convert scan-*jpg scan.ps
$ ls -l scan*
-rw-r--r-- 1 cromwell cromwell   131879 Nov 22 10:34 scan-0001.jpg
-rw-r--r-- 1 cromwell cromwell   137088 Nov 22 10:35 scan-0002.jpg
-rw-r--r-- 1 cromwell cromwell   132204 Nov 22 10:35 scan-0003.jpg
-rw-r--r-- 1 cromwell cromwell   133016 Nov 22 10:36 scan-0004.jpg
-rw-rw-r-- 1 cromwell cromwell   479604 Nov 22 11:41 scan.pdf
-rw-rw-r-- 1 cromwell cromwell 13517838 Nov 22 11:41 scan.ps
```

However, fixing this problem has shown us other PostScript-based security problems.

Further Postscript and PDF Risks on Desktops

You can search for software *trying* to use Ghostscript safely by calling it with the `-dSAFER` flag, which we realize doesn't work. It's a long list:

a2ps, auctex, auto-07p, boost1.62, boost1.63, boost1.67, c2050, cairo, calligra, camlimages, cenon.app, chezscheme, claws-mail, context, courier, cups-filters, cups-pdf, cups-x2go, derivations, djtools, efax, efax-gtk, emacs, emacs25, evolution, fbi, fig2dev, fig2ps, fim, foo2zjs, foomatic-db-engine, freebsd-smbfs, gdal, ghostscript, gimp, gle-graphics, gmt, gramps, graphicsmagick, groff, gv, hplip, ifhp, imagemagick, impose+, k2pdfopt, kde4libs, klatexformula, latex2rtf, latexdiff, leafnode, libpodofo, libpostscriptbarcode, lilypond, luasocket, lyx, m2300w, magicfilter, magics++, mgetty, min12xxw, mupdf, nltk, nx-libs, ocrmypdf, octave, plastex, pnm2ppa, printfilters-ppd, ps2eps, pstotext, python-uniconvertor, pyxplot, r-cran-tm, racket, ruby-asciidoctor-pdf, ruby-tioga, stex, texlive-base, texlive-bin, texlive-extra, texlive-lang, texstudio, texworks-manual, vim, webhelpers, xdvik-ja, xemacs21-packages, xfig, xutils-dev, yorick, yorick-mira, yorick-yutils

Some are obvious, like components of LaTeX and print service. Other, though, are somewhat cryptic. There are some dangerous ones lurking in the list.

Desktop environments like Gnome and KDE include file indexing and search tools. These automatically run in the background, reading *all* of your files with a variety of software packages, including the dangerous PostScript interpreters.

If you're wondering why your desktop runs slowly, and why the disk activity light is so very busy, it's likely because of these indexing tools!

Also, a graphical file manager may use PostScript interpreters to create previews or "thumbnails".

Unfortunately, all of these may have to be disabled or reconfigured for each user. Depending on your distribution, there *may* be a system-wide configuration file under `/usr/share/kde-settings/kde-profile/default/share/config/`. And, we can't simply remove the packages due to dependencies.

Your distribution might have put things in strange places. So, before continuing you might want to find where configuration files might have been hidden.

```
$ find .??* -type f | sort | egrep 'baloo|nepomuk|akonadi|strigi|dolphin'
```

Disabling Nepomuk and Strigi Indexing Services

Edit `~/.kde*/share/config/nepomukserverrc` and change any `start` or `autostart` values from `true` to `false`.

```
[... lines deleted ...]

[Basic Settings]
Start Nepomuk=false

[Service-nepomukfileindexer]
autostart=false

[Service-nepomuktelepathyservice]
autostart=false

[... lines deleted ...]
```

Disabling Baloo Indexing Service

Edit `~/.kde*/share/config/baloofilerc` or `~/.config/baloofilerc` and change `Indexing-Enabled` to `false`.

```
[... lines deleted ...]

[Basic Settings]
Indexing-Enabled=false

[... lines deleted ...]
```

Disabling Akonadi Indexing Service

Edit `~/kde*/share/config/akonadiserverrc` or `~/.config/akonadi/akonadiserverrc` and change `StartServer` to `false`.

```
[... lines deleted ...]
StartServer=false
```

Disabling PostScript/PDF Previews in the Dolphin File Manager

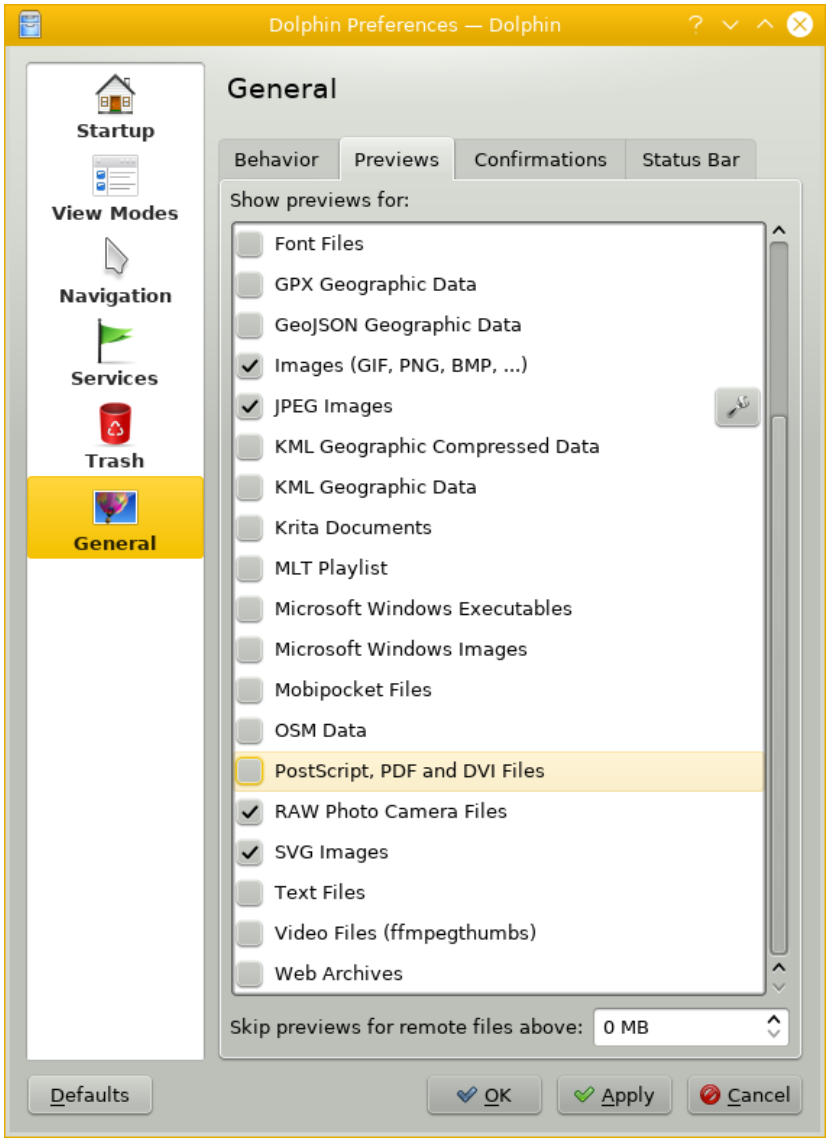
Edit `~/.kde*/share/config/dolphinrc` or `.config/dolphinrc`. In the `[PreviewSettings]` stanza, `Plugins` should *not* include `gsthumbnail`

```
[... lines deleted ...]

[PreviewSettings]
Plugins=directorythumbnail,imagethumbnail,jpegthumbnail,rawthumbnail,svgthumbnail

[... lines deleted ...]
```

Or, you can change it through its menus.
Control → Configure Dolphin → General → Previews:



Back to the Linux / Open-Source Page

Home
Travel
Linux/Unix
Cybersecurity
Networking
Technical
Radio
Site Map