Preservation Planning Module - Bug #5941

Fido script has problem identifying Powerpoint files

11/08/2013 07:01 AM - Justin Simpson

Status:	Verified	**Start date:**	11/08/2013
Priority:	Medium	**Due date:**	
Assignee:	Sarah Romkey	**% Done:**	100%
Category:	FPR Rules Database	**Estimated time:**	0.00 hour
Target version:	Release 1.4.0		
Sponsored:	No	**Requires documentation:**	
Pull Request:			

Description

When using FIDO for File Identification, on a file from the testTransfers maildir set (PPT_test.ppt), FIDO identifies the file as fmt/111, which is a generic Microsoft OLE2 Compound Document Format. Identifying by file extension (.ppt) achieves a more accurate result in this case.

This is not really a bug, FIDO is correctly identifying the file as microsoft document, it just can't tell the difference between a word file and a powerpoint or excel. Possible enhancement would be to have the script that calls FIDO look at file extension for fmt/111 results.

Example output:

```
IDCommand UUID: 15dd336a-fa0a-45ba-b9c1-63c3f4bc8b8b
File: (49f7a721-a607-4b82-9149-61e3008f395d)
/var/archivematica/sharedDirectory/watchedDirectories/workFlowDecisions/selectFormatIDToolTransfer/maildir_test4-eb050832-44b
4-442b-88dd-82335292244b/objects/attachments/_Gmail_.All_Mail/cur/587c6220-e03e-421c-b573-c0dc91b820b5_Project.zip-2013-
11-08T17_28_55.631012/PPT_test.ppt
fmt/111

Command output: fmt/111
```

History

#1 - 11/12/2013 07:33 AM - Misty De Meo

Looks like the issue is that the specific PowerPoint header isn't until fairly far into the file. Fido's default buffer size is 128KB, so it will use the most specific header it finds within that buffer region. Increasing the buffer size to 1MB (via passing `-bufsize 1048576`) allows the file to be identified correctly, at the cost of increasing execution time.

In the case of PoewrPoint files, execution goes from an average of about 230ms to 520ms. Other file formats also have increased scanning time, though not quite as severe.

#2 - 11/12/2013 08:05 AM - Misty De Meo

This isn't the only fido issue related to the default buffer size (see #5731), so it's probably worth eating the extra execution time if it means we identify popular formats correctly.

#3 - 11/14/2013 02:37 PM - Misty De Meo

The FIDO script should be updated to the following:

```
import os.path
import re
import subprocess
import sys
```

```
def file_tool(path):
    return subprocess.check_output(['file', path]).strip()

class FidoFailed(Exception):
    def __init__(self, stdout, stderr, retcode):
        message = """
Fido exited {retcode} and no format was found.
stdout: {stdout}
---
stderr: {stderr}
""".format(stdout=stdout, stderr=stderr, retcode=retcode)
        super(FidoFailed, self).__init__(message)

def identify(file_):
    # The default buffer size fido uses, 256KB, is too small to be able to detect certain formats
    # Formats like office documents and Adobe Illustrator .ai files will be identified as other, less-specific formats
    # This larger buffer size is a bit slower and consumes more RAM, so some users may wish to customize this to reduce the buffer size
    # See: https://projects.artefactual.com/issues/5941, https://projects.artefactual.com/issues/5731
    cmd = ['fido', '-bufsize', '1048576', os.path.abspath(file_)]
    process = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
    stdout, stderr = process.communicate()

    try:
        results = stdout.split('\n')[0].split(',')
    except:
        raise FidoFailed(stdout, stderr, process.returncode)

    if process.returncode != 0 or results[-1] == '"fail"':
        raise FidoFailed(stdout, stderr, process.returncode)
    else:
        puid = results[2]
        if re.match('(.+)?fmt\/\d+', puid):
            return puid
        else:
            print >> sys.stderr, "File identified as non-standard Fido code: {id}".format(id=puid)
            return ""

def main(argv):
    try:
        print identify(argv[1])
        return 0
    except FidoFailed as e:
        file_output = file_tool(argv[1])
        # FIDO can't currently identify text files with no extension, and this
        # is a common enough usecase to special-case it
        if 'text' in file_output:
            print 'x-fmt/111'
        else:
            return e
    except Exception as e:
        return e

if __name__ == '__main__':
    exit(main(sys.argv))
```

Once the FPR is updated, we should update the documentation somewhere so that the decision is made clear to users.

#4 - 11/15/2013 01:22 PM - Misty De Meo

- Status changed from New to QA/Review

- % Done changed from 0 to 100

Updated on the QA server and tested. We should still update the documentation somehow to alert users, though the fido command has a comment noting this.

#5 - 04/29/2015 11:33 AM - Justin Simpson

- Status changed from QA/Review to Document

- Assignee changed from Misty De Meo to Sarah Romkey

- Target version changed from Release 1.1.0 to Release 1.4.0

We should verify that this is documented correctly for the Archivematica 1.4.0 release.

#6 - 06/29/2015 01:01 PM - Sarah Romkey

- Status changed from Document to Verified

Added note in Identification section of Preservation Planning:

https://www.archivematica.org/en/docs/archivematica-1.4/user-manual/preservation/preservation-planning/#identification