

Fuzzer

This README file describe how to use our tools, including TypeOracle, Cooper+TypeOracle and Favocado+TypeOracle, to fuzz PDF Readers.

TypeOracle

This fuzzer uses the type information reasoned by TypeOracle to fuzz PDF Readers

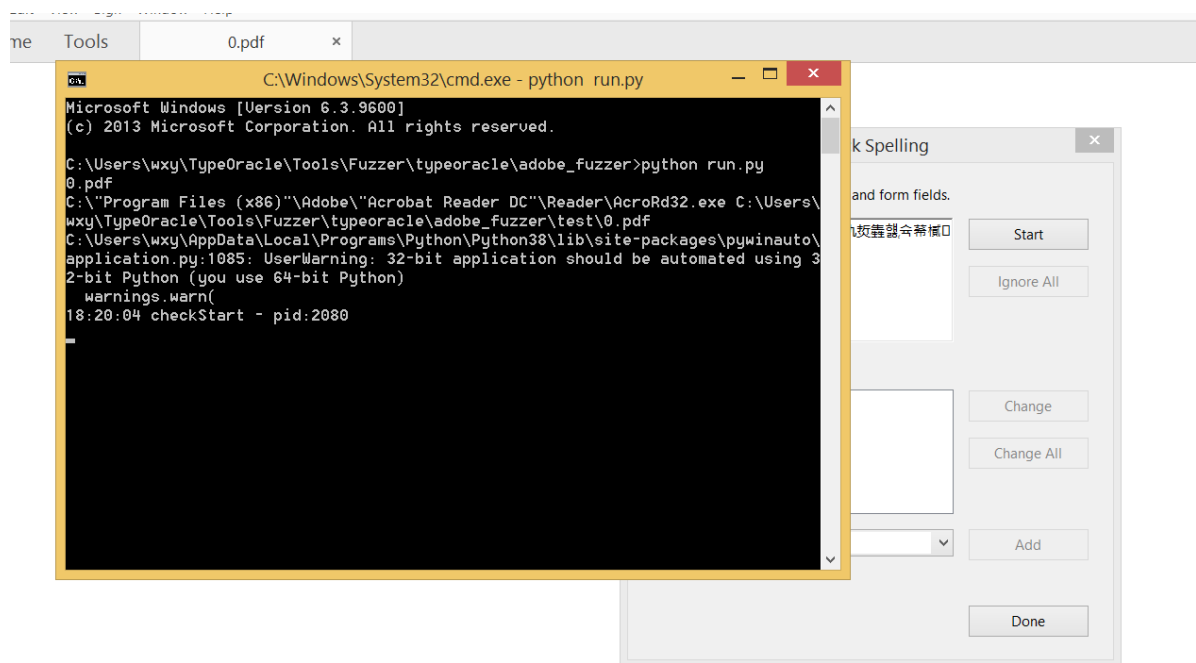
folder structure

- config
 - blacklist.txt (binding calls that don't test)
 - constant.txt (some constant variables)
 - delist.txt (binding calls that test less)
 - objlst.txt (list of built-in objects)
 - strlst.json (some special strings)
 - template.pdf (PDF template)
- data (the folder stores the type information of binding calls)
- save (save the test PDFs that trigger unormal behavior)
- test (save all test PDFs)
- generate.py (generate Javascript codes)
- monitor.py (monitor and interact with PDF Reader)
- mPDF.py (embeded Javascript codes to PDFs)
- pattern.py (parse type information file of every binding call)
- rand.py (generate random value)
- relationship.json (some binding calls that have relationships)
- run.py

how to reproduce

1. execute run.py to start fuzz

```
python run.py
```



2. results are stored in `save` folder

```
crash: PDFs make PDF Reader crash
halt: PDFs make PDF Reader exit without any notice
hang: PDFs make PDF Reader hang
stop: PDFs make PDF Readerstop execute Javascript codes
error: PDFs make our fuzz make mistakes
```

Cooper+TypeOracle

The fuzzer use Cooper integrated with TypeOracle's type information to generate test PDFs, and use our monitor tool to monitor and interact with PDF Reader. The source code of Cooper can be found at <https://github.com/TCA-ISCAS/Cooper>.

folder structure

- config
- Cooper
- data (the folder stores the type information of binding calls)
- PdfData (result that parse PDF's of Cooper)
- PdfJsTemplate (Cooper's original Javascript generation template)
- PdfPaser (Cooper's utility to parse PDF's objects)
- PdfSamples (sample provided by Cooper's authors)
- save (save the test PDFs that trigger unormal behavior)

- test (save all test PDFs)
- generate.py (generate Javascript codes)
- grammar.py
- monitor.py (monitor and interact with PDF Reader)
- myPDF.py (embeded Javascript codes to PDFs)
- pattern.py (parse type information file of every binding call)
- PdfExecutionLog.py
- PdfMutation.py
- PdfSolution.py
- rand.py (generate random value)
- relationship.json (some binding calls that have relationships)
- run.py

how to reproduce

1. use python2 to execute run.py to launch fuzzer

```
C:\python27-x64\python run.py
```

```
C:\Windows\System32\cmd.exe
C:\Users\wxu\TupeOracle\Tools\Fuzzer\cooper_tupeoracle\adobe\PdfPaser\pdf.py:484
hon run.py
pdf_cooper_data_prepare return
Mutate api:OCGS
Mutate api:dataObjects
Mutate api:Field
Mutate api:Annotation
cooper_guided_mutation end
[(u'OCGS', 'AUbfEK3cRI+eAT-u5KxccQ==.pdf', [((134, '/Resources'), 38)]), (u'OCGS', 'Ge1RDj-wactjez9kgJ0TJA==.pdf', [((1030, 36), ((2, 63))], (u'OCGS', 'YP-K26ifS0rWqgnq+FE3AQ==.pdf', [((92, 69), ((115, 69))], (u'dataObjects', 'RNSsyCTFX32Jl-IK1JQ2dg==.pdf', [((18, 74)]), (u'dataObjects', 'joHQwMXyZSnFXIHauXdrma==.pdf', [((20, '/Resources', '/ColorSpace'), 48), ((19, '/Resources', '/ColorSpace'), 48), ((16, '/Resources', '/ExtGState'), 59)]), (u'dataObjects', 'pw-1GpFmHhp2PY8rqARxKg==.pdf', [((7824, '/CIDSystemInfo'), 49)]), (u'Field', 'fk1CaGaN49ePHjMbyNHH9A==.pdf', [((0, '/MarkInfo'), 41)]), (u'Field', 'Spcak26DhMzKqrDsuwUNzg==.pdf', [((86, 13), ((91, 13))], (u'Field', 'XQyIgJuHWFng3xA0kBKEeQ==.pdf', [((765, 34), ((738, 49), ((771, 34))], (u'Annotation', 'S0s1UNF+5Q0ShfGhIiz3ug==.pdf', [((258, 35)]), (u'Annotation', 'qGJwomS0-rpml6b9h0Ur9w==.pdf', [((820, 34)])])
C:\Users\wxu\TupeOracle\Tools\Fuzzer\cooper_tupeoracle\adobe\PdfPaser\pdf.py:484
: PdfReadWarning: Xref table not zero-indexed. ID numbers for objects will be corrected.
utils.PdfReadWarning)
C:\Users\wxu\TupeOracle\Tools\Fuzzer\cooper_tupeoracle\adobe\PdfPaser\pdf.py:484
```

2. results are stored in `save` folder

```
crash: PDFs make PDF Reader crash
halt: PDFs make PDF Reader exit without any notice
hang: PDFs make PDF Reader hang
stop: PDFs make PDF Reader stop execute Javascript codes
error: PDFs make our fuzz make mistakes
```

Favocado+TypeOracle

The fuzzer use Favocado integrated with TypeOracle's type information to generate test PDFs, and use our monitor tool to monitor and interact with PDF Reader. The source code of Favocado can be found at <https://github.com/favocado/Favocado>.

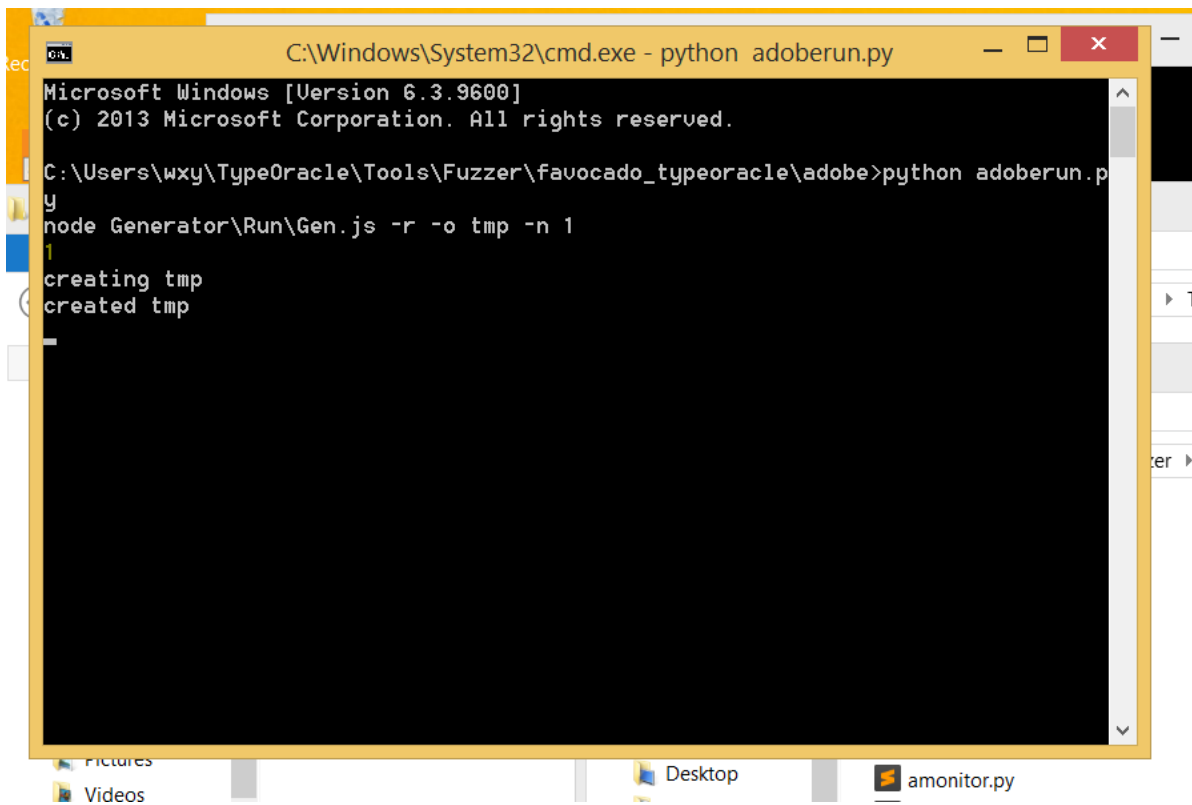
folder structure

- Generator (folders that store the major part of favocado, integrating with our's type information)
- save (save the test PDFs that trigger unormal behavior)
- test (save all test PDFs)
- tmp (save the javascript codes)
- adoberun.py/foxitrun.py (launch the fuzzer)
- amonitor.py/fmonitor.py (monitor and interact with PDF Reader)
- generate.py (execute the command to generate javascript codes)

how to reproduce

1. execute adoberun.py/foxitrun.py

```
python adoberun.py
python foxitrun.py
```



```
C:\Windows\System32\cmd.exe - python adoberun.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\wxy\TypeOracle\Tools\Fuzzer\favocado_typeoracle\adobe>python adoberun.py
y
node Generator\Run\Gen.js -r -o tmp -n 1
1
creating tmp
created tmp
```

2. results are stored in `save` folder

```
crash: PDFs make PDF Reader crash
halt: PDFs make PDF Reader exit without any notice
hang: PDFs make PDF Reader hang
stop: PDFs make PDF Readerstop execute Javascript codes
error: PDFs make our fuzz make mistakes
```