



MARCH 28-31, 2017


MARINA BAY SANDS / SINGAPORE

Dig Into The Attack Surface of PDF and Gain 100+ CVEs in 1 Year

Ke Liu

Tencent's Xuanwu Lab

About Me

- Ke Liu
 - From Beijing, China
 - Security researcher of Tencent's Xuanwu Lab
 - Focus on file format fuzzing (PDF, Images, etc.)
 - Twitter: @klotxl 



Agenda

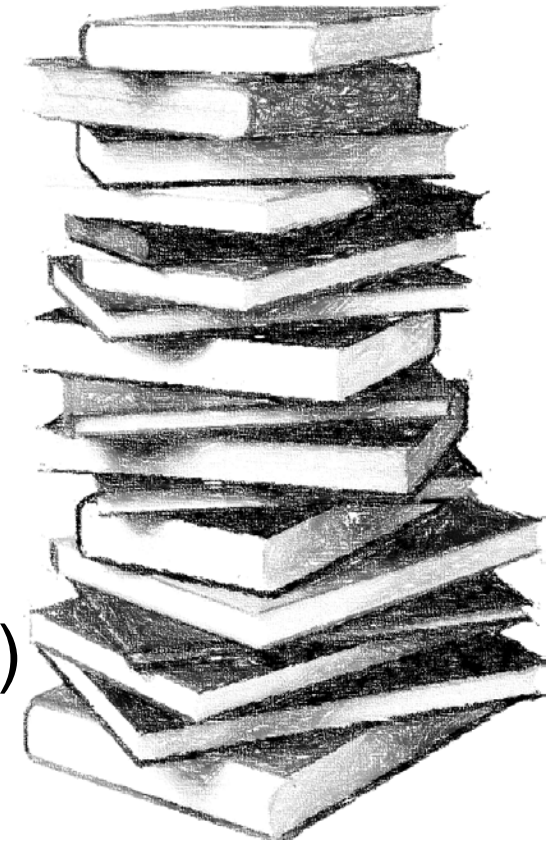
- Why Portable Document Format
- Find the Attack Surface
- Get More Test Cases
- Fuzzing Tricks
- Statistical Results

Why Portable Document Format

- Previous researches about PDF security
 - **Nicolas Grégoire**: Fuzzing XSLT Engine
 - **Zero Day Initiative**: Abusing JavaScript APIs
 - **Sebastian Apelt**: Pwning XFA Engine
 - etc.
- We can do more!
 - PDF is a complex format, more than you can imagine.

Why Portable Document Format

- PDF is a complex format
 - PDF Specification (**756 pages**)
 - JavaScript for Acrobat (**769 pages**)
 - XFA Specification (**1584 pages**)
 - Fonts (TrueType, Type0, Type1, Type3, etc.)
 - Images (Jpeg2000, Jpeg, Png, Bmp, Tiff, Gif, etc.)
 - Other features (XSLT, etc.)



Make PDF Secure Again

- Patched vulnerabilities (since December 2015)

CVE-2016-0947, CVE-2016-1118, CVE-2016-1119, CVE-2016-1120, CVE-2016-1121, CVE-2016-1122, CVE-2016-1123, CVE-2016-1124
CVE-2016-1125, CVE-2016-1126, CVE-2016-1127, CVE-2016-1128, CVE-2016-1129, CVE-2016-1130, CVE-2016-1685, CVE-2016-1686
CVE-2016-2401, CVE-2016-3203, CVE-2016-3617, CVE-2016-3618, CVE-2016-4088, CVE-2016-4089, CVE-2016-4090, CVE-2016-4091
CVE-2016-4092, CVE-2016-4093, CVE-2016-4094, CVE-2016-4095, CVE-2016-4096, CVE-2016-4097, CVE-2016-4098, CVE-2016-4099
CVE-2016-4100, CVE-2016-4101, CVE-2016-4102, CVE-2016-4103, CVE-2016-4104, CVE-2016-4105, CVE-2016-4106, CVE-2016-4107
CVE-2016-4193, CVE-2016-4194, CVE-2016-4201, CVE-2016-4203, CVE-2016-4208, CVE-2016-4209, CVE-2016-4210, CVE-2016-4211
CVE-2016-4212, CVE-2016-4213, CVE-2016-4214, CVE-2016-4250, CVE-2016-4254, CVE-2016-4256, CVE-2016-4257, CVE-2016-4258
CVE-2016-4259, CVE-2016-4260, CVE-2016-4261, CVE-2016-4262, CVE-2016-4660, CVE-2016-4671, CVE-2016-4681, CVE-2016-5140
CVE-2016-5210, CVE-2016-6229, CVE-2016-6861, CVE-2016-6862, CVE-2016-6863, CVE-2016-6864, CVE-2016-6865, CVE-2016-6993
CVE-2016-6994, CVE-2016-6995, CVE-2016-6996, CVE-2016-6997, CVE-2016-6998, CVE-2016-6999, CVE-2016-7000, CVE-2016-7001
CVE-2016-7002, CVE-2016-7003, CVE-2016-7004, CVE-2016-7005, CVE-2016-7006, CVE-2016-7007, CVE-2016-7008, CVE-2016-7009
CVE-2016-7010, CVE-2016-7011, CVE-2016-7013, CVE-2016-7014, CVE-2016-7015, CVE-2016-7016, CVE-2016-7017, CVE-2016-7018
CVE-2016-7019, CVE-2016-7852, CVE-2016-7853, CVE-2016-8875, CVE-2016-8876, CVE-2016-8877, CVE-2016-8878, CVE-2016-8879
CVE-2017-2940, CVE-2017-2942, CVE-2017-2943, CVE-2017-2944, CVE-2017-2945, CVE-2017-2952, CVE-2017-2953, CVE-2017-2954
CVE-2017-2959, CVE-2017-2960, CVE-2017-2963, CVE-2017-2964, CVE-2017-2965, CVE-2017-2966, CVE-2017-2972, CVE-2017-5556

Find the Attack Surface

The Standard Documents
(Thousands of pages)

Installation Files
(Adobe Reader)

How to find?

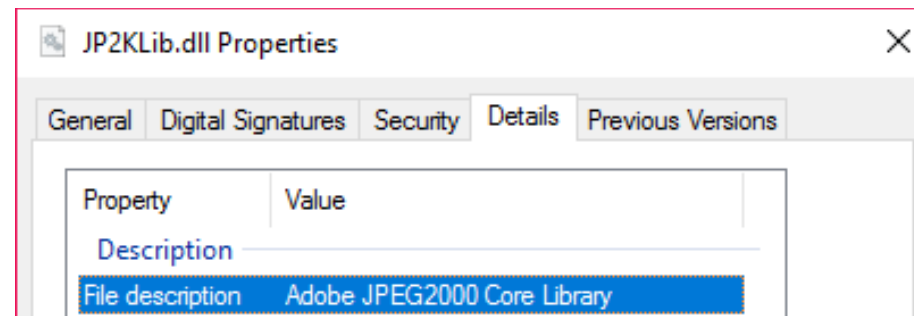
Open Source Projects
(PDFium)

Security Advisories
(APSB, ZDI, Chromium)

Find the Attack Surface

- Installation Files Investigation

- File name
- Properties
- Strings
- Function names
- Copyrights information
- etc.



Property	Value
Description	
File description	Adobe JPEG2000 Core Library
Type	Application extension
File version	1.2.2.37137
Product name	JP2KLib 2016/09/08-12:02:14
Product version	84.264258
Copyright	© 2016 Adobe Systems Incorporated. All rights reserved.
Size	1,048,576 bytes
Date modified	9/8/2016 12:02:14 AM
Language	English (United States)
Legal trademarks	Adobe ®
Original filename	JP2KLib.dll

JPEG2000

Find the Attack Surface

- Installation Files Investigation (Adobe Reader)

File Name	Description
AcroRd32.dll	Core Library
ACE.dll	Color Engine
AGM.dll	Graphics Manager
AXSLE.dll	XSLT Engine
CoolType.dll	Font Engine
JP2KLib.dll	JPEG2000 Codec Library
JSByteCodeWin.bin	JavaScript Functions

Find the Attack Surface

- Plugin Files Investigation (Adobe Reader)

File Name	Description
AcroForm.api	Acrobat Forms Plugin (XFA)
Annots.api	Annotation Plugin
EScript.api	JavaScript Engine
ImageConversion.api	Image Converter (Acrobat only)
Multimedia.api	Multimedia Plugin
PPKLite.api	Public-Key Security Plugin
weblink.api	WebLink Plugin

Find the Attack Surface

- Open Source Projects (PDFium)
 - Source code analysis
 - libFuzzer components ([master/testing/libfuzzer](#))

Attack Surface	libFuzz component (filename prefix: pdf_)
XML	cfx_saxreader_fuzzer.cc, xml_fuzzer.cc
Font	cmap_fuzzer.cc
Image Codec	codec_(*)_fuzzer.cc (bmp, fax, gif, icc, jbig2, jpeg, png, tiff), jpx_fuzzer.cc
Engine	hint_table_fuzzer.cc, streamparser_fuzzer.cc, psengine_fuzzer.cc
XFA Forms	css_fuzzer.cc, fm2js_fuzzer.cc

Get More Test Cases

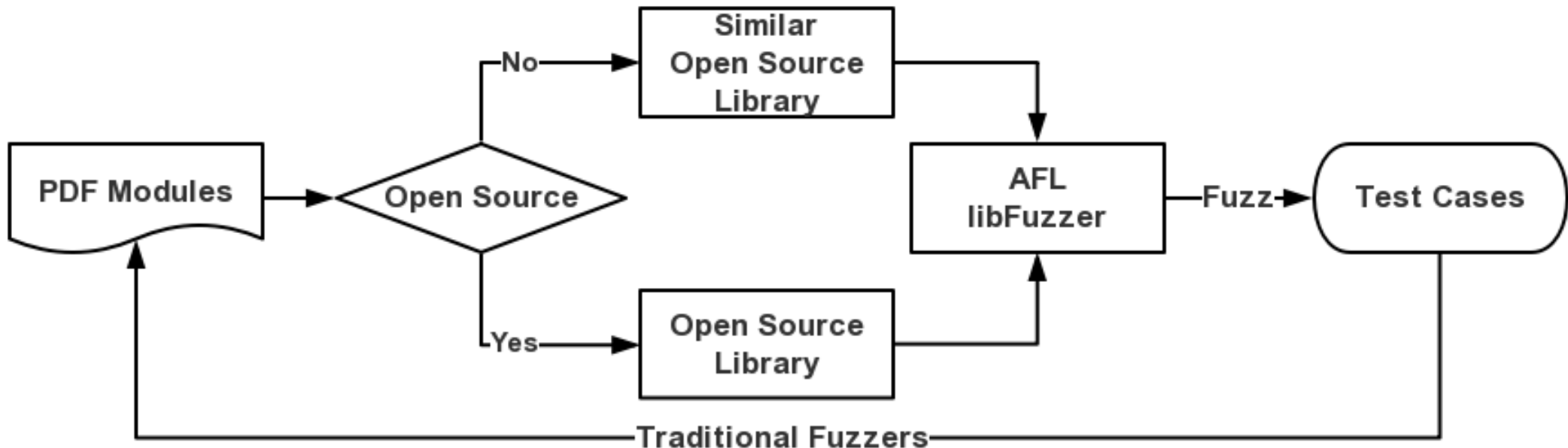
- The importance of test cases
 - More **test cases** mean more possible **code coverages**.
 - More **code coverages** mean more possible **crashes**.
- How to get more test cases?
 - Writing a crawler is acceptable in most cases. But there are some alternative ways.

Get More Test Cases

- Strategies for modern fuzzers (AFL & libFuzzer)
 - AFL: A single good and small test case is enough
 - libFuzzer:
 - A (**minimized**) set of test cases
 - Can work without any initial test cases
- Common features
 - Generate lots of test cases to get higher code coverage rate

Get More Test Cases

- Don't waste your test cases
 - Fuzzing open source libraries with AFL / libFuzzer
 - Testing binaries with test cases generated in previous steps



Get More Test Cases

- Get help from open source projects
 - Popular project also maintains a test case repository which contains lots of valid and invalid files.

Project	Format	Test Case Repository
PDFium	PDF	https://pdfium.googlesource.com/pdfium_tests/ https://pdfium.googlesource.com/pdfium/testing/resources/
PDF.js	PDF	https://github.com/mozilla/pdf.js/tree/master/test/pdfs
OpenJPEG	JPEG2000	https://github.com/uclouvain/openjpeg-data
LibTIFF	TIFF	ftp://download.osgeo.org/libtiff
Google Noto Fonts	TTF	https://www.google.com/get/noto/

Fuzzing Tricks

- Write PDF makers
 - Why?
 - Only mutate the data that you're interested
 - How?
 - Use third-party PDF makers or libraries is a choice, but error checking functionality may lose lots of test cases.
 - Alternatively, read the standard documents and write your own PDF makers.

Fuzzing Tricks

- Fuzz third-party libraries
 - Check if the target uses open source libraries
 - Fuzz open source library with AFL/libFuzzer is more efficient
 - Target may be affected by known vulnerabilities
 - Zero day vulnerabilities affect all targets that use the library

Fuzzing Tricks

- Case study: LibTIFF
 - CVE-2016-5875 LibTIFF OOB Write in PixarLogDecode
 - I discovered it in late June last year
 - Also discovered by other researchers

Product	Affected	Remark
Adobe Acrobat Pro DC	Yes 😊	ImageConversion Plugin
Google Chrome	Yes 😊	Dev and Beta with XFA enabled
Foxit Reader	Yes 😊	Rendering engine and ConvertToPDF Plugin
Adobe Reader DC	No 😞	PixarLog compression not supported



AcroForm.api
API File
12.6 MB

```
"%s compression support is not configured"
```

Fuzzing Tricks

- Why bother fuzzing PDF Readers / Browsers?
 - Creating reader / browser process is time consuming
 - Loading lots of unnecessary libraries
 - Doing lots of unnecessary initializations
 - GUI is also unnecessary under some circumstances
- Write wrappers
 - Load, initialize, and execute unnecessary code as less as possible. Focus on what you're interested.

Fuzzing Tricks

- Write wrappers
 - Target is open source (PDFium)
 - Target has SDK (Foxit Reader)
 - Target has APIs (Windows PDF Library)
 - Reverse a proper function in some libraries (Adobe Acrobat)

Fuzzing Tricks

- Case study: Windows PDF Library
 - Microsoft ships Windows.Data.PDF.dll since Windows 8.1
 - Can be interacted through Windows Runtime APIs

IPdfDocumentStatics->LoadFromStreamAsync



IPdfDocument->GetPage



IPdfPage->PreparePageAsync & RenderToStreamAsync

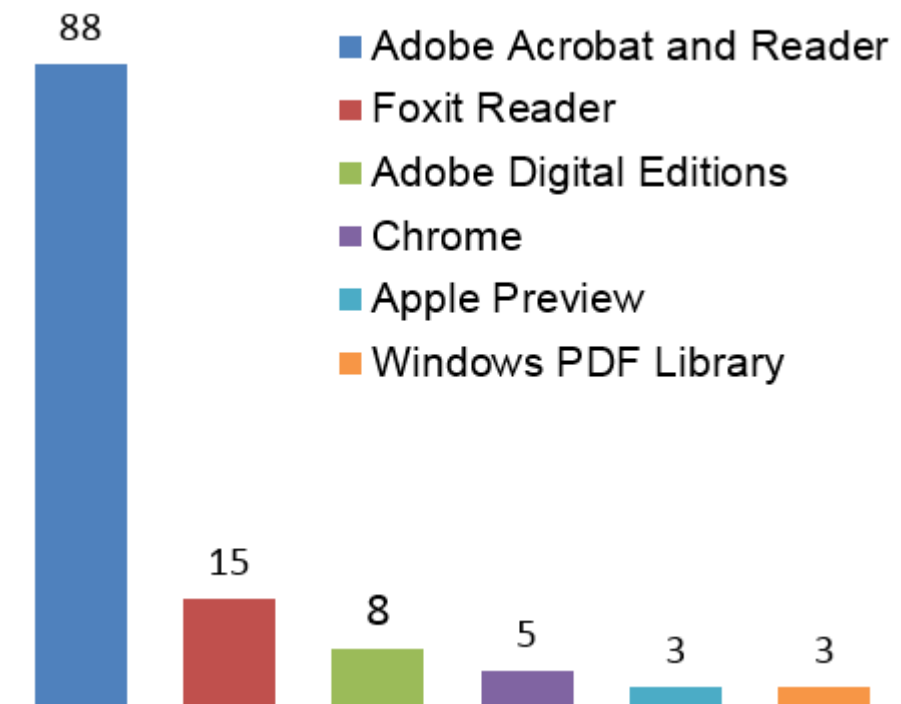
Statistical Results

- About the research
 - Started since December 2015
 - 122 vulnerabilities (collisions excluded) have been patched
 - Dozens of unpatched vulnerabilities on the way
- Vendors



Statistical Results

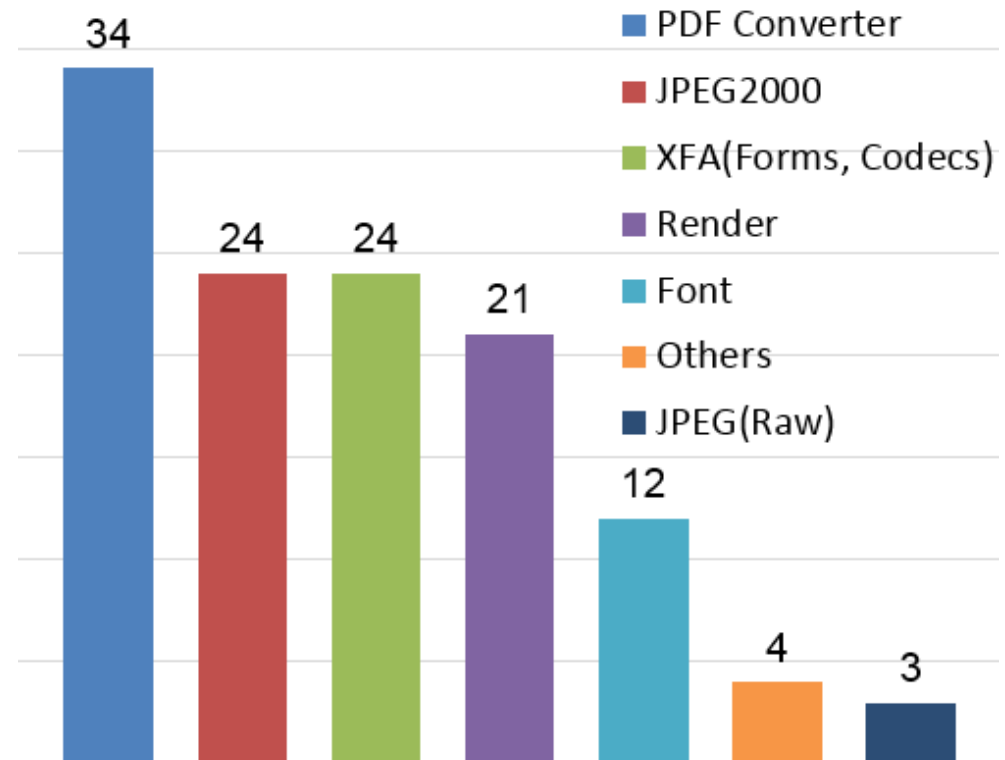
- Vulnerability distribution by vendor
 - Research mainly focus on Adobe Acrobat and Reader
 - One CVE for each vulnerability
 - Vulnerability collisions excluded
 - Beta, Dev versions excluded
 - Only patched ones counted
 - Results just for reference



Statistical Results

- Vulnerability distribution by attack surface

- Research mainly focus on Adobe Acrobat and Reader
- One CVE for each vulnerability
- Vulnerability collisions excluded
- Beta, Dev versions excluded
- Only patched ones counted
- Results just for reference



Statistical Results

- A short story about Chrome
 - **2016/05/16**: XFA was enabled in Chrome Canary
 - **2016/05/17**: XFA was enabled in Chrome Dev
 - **2016/06/01**: Started to fuzz and reported dozen of issues
 - **2016/06/06**: Official fuzzers were added
 - **2016/06/08**: XFA was enabled in Chrome Beta
 - **2016/06/15**: XFA was disabled in all versions

Black Hat Sound Bytes

- Finding the attack surface of PDF
 - The attack surface of PDF and how to find it
- Collecting test cases
 - From open source projects and modern fuzzers
- Fuzzing tricks
 - Make your fuzzer more efficiently

Thanks for listening

