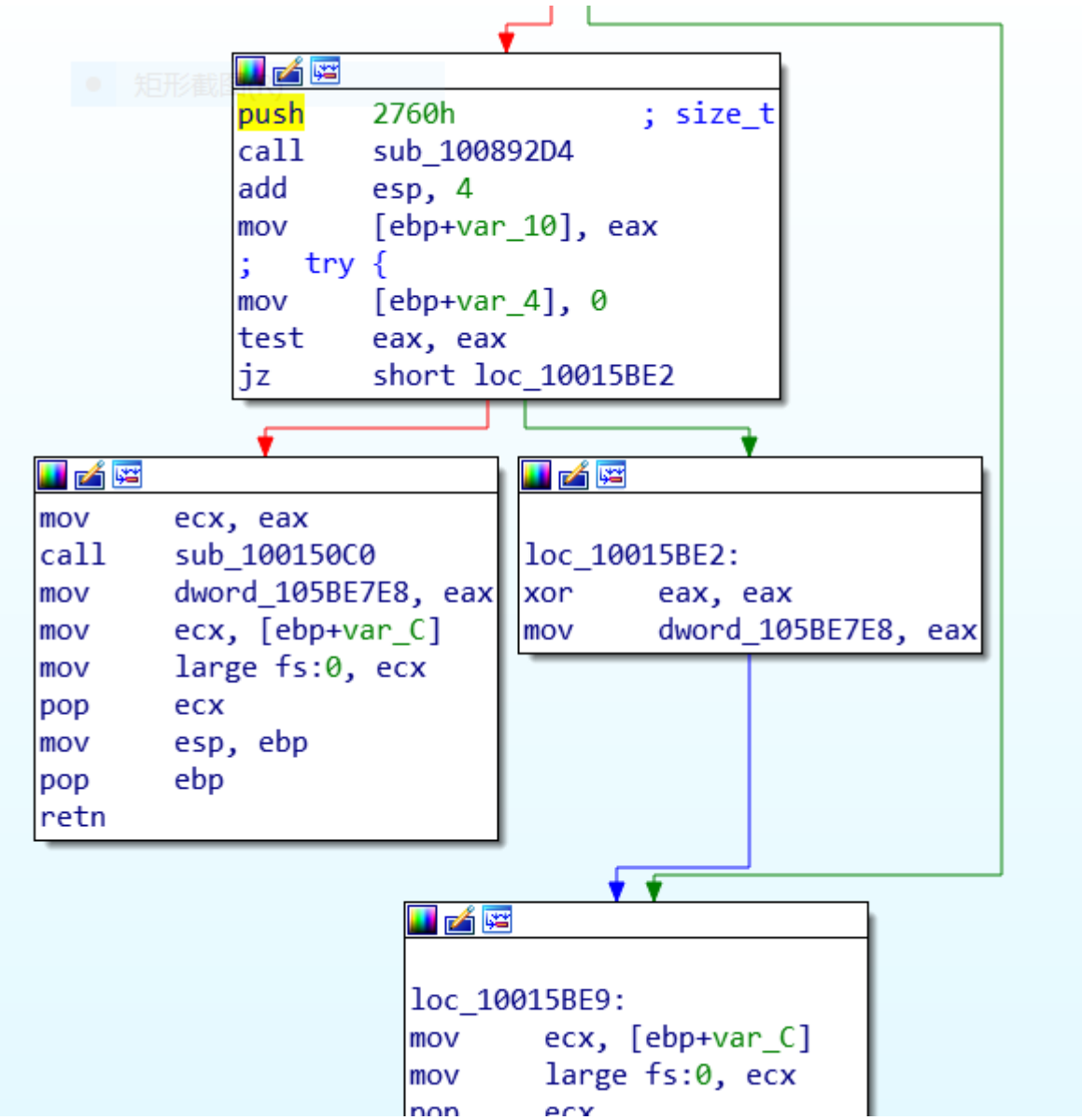
试写foxit reader的ConvertToPDF功能的wrapper

cnblogs.com/st404/p/9384704.html

相比于直接fuzzing大型程序本身，针对程序的某一特定功能写wrapper后再fuzzing则要高效的多。网上搜了下，仅有两篇关于foxit reader的wrapper文章，一个用python，另外一个用C++，而且针对的foxit reader版本也比较旧。本篇的目的通过分析C++的wrapper原理，来写出最新版foxit reader（Version: 9.1.0.5096）的ConvertToPDF功能的wrapper。

首先看下ConvertToPDF_x86.dll插件的反汇编部分



刚开始分配0x2760h大小的内存，然后可以看到

```
.text:10015BC7              mov     ecx, eax
.text:10015BC9              call    sub_100150C0
```

从这两句可以猜测ConvertToPDF_x86.dll插件中存在虚函数，因为ecx中存储有类实例的this指针，它作为隐藏的第一个参数传递给sub_100150C0。具体原理可参考http://www.openrce.org/articles/full_view/23。然后继续跟进sub_100150C0函数，如下

```
mov     esi, ecx
mov     [ebp+var_10], esi
lea     ecx, [esi+4]    ; void *
mov     dword ptr [esi], offset ??_7CFX_PDFConvertor@@6B@ ; const CFX_PDFConvertor::`vftable'
```

根据这段代码可以确定esi中存储有虚函数表的首地址，虚函数表的具体函数如下：

```
64dd020c  0000000a
0:000:x86> r
eax=04afb760 ebx=00000000 ecx=153bb91c edx=153b024c esi=153bb918 edi=091f3360
eip=64a750f3 esp=04afb750 ebp=04afb76c iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200206
ConvertToPDF_x86!CreateFXURLToHtml+0x1633:
64a750f3 e898faffff      call    ConvertToPDF_x86!CreateFXURLToHtml+0x10d0 (64a74b90)
0:000:x86> dds poi(esi)
64dd0190  64a75c10 ConvertToPDF_x86!CreateFXPDFConvertor+0x90
64dd0194  64a73f10 ConvertToPDF_x86!CreateFXURLToHtml+0x450
64dd0198  64a74750 ConvertToPDF_x86!CreateFXURLToHtml+0xc90
64dd019c  64a72fa0 ConvertToPDF_x86+0x12fa0
64dd01a0  64a76190 ConvertToPDF_x86!CreateFXPDFConvertor+0x610
64dd01a4  64a747c0 ConvertToPDF_x86!CreateFXURLToHtml+0xd00
64dd01a8  64a73dd0 ConvertToPDF_x86!CreateFXURLToHtml+0x310
64dd01ac  64a75200 ConvertToPDF_x86!CreateFXURLToHtml+0x1740
64dd01b0  00460043
64dd01b4  005f0058
64dd01b8  00440050
64dd01bc  00430046
```

ConvertToPDF_x86.dll插件的主要构成函数如上图，我们只需函数之间的确定执行流程、每个函数的主要参数内容及个数即可完成wrapper。

主要的函数流程依次为

ConvertToPDF_x86!CreateFXURLToHtml+0x450

ConvertToPDF_x86!CreateFXURLToHtml+0xc90

ConvertToPDF_x86!CreateFXPDFConvertor+0x90

参数个数的确定，因为函数的调用约定遵循thiscall，所以每次调用完函数后，由被调用函数自动清除函数参数，具体代码为ret xx

```
64dd020c  0000000a
0:000:x86> bp ConvertToPDF_x86!CreateFXURLToHtml+0x450
0:000:x86> g
Breakpoint 1 hit
ConvertToPDF_x86!CreateFXURLToHtml+0x450:
64a73f10 55              push    ebp
0:000:x86> uf ConvertToPDF_x86!CreateFXURLToHtml+0x450
ConvertToPDF_x86!CreateFXURLToHtml+0x450:
64a73f10 55              push    ebp
64a73f11 8bec            mov     ebp,esp
64a73f13 6aff            push    0FFFFFFFFh
64a73f15 68a804dc64      push    offset
ConvertToPDF_x86!ConnectedPDF::ConnectedPDFSDK::FCP_SendEmailNotification+0x2cc28
(64dc04a8)
64a73f1a 64a100000000    mov     eax,dword ptr fs:[00000000h]
64a73f20 50              push    eax
64a73f21 83ec14          sub     esp,14h
64a73f24 53              push    ebx
```

。。。。。。。。。。。。。。。。。。。。

```
ConvertToPDF_x86!CreateFXURLToHtml+0xb70:
64a74630 33c0            xor     eax,eax
64a74632 8b4df4          mov     ecx,dword ptr [ebp-0Ch]
64a74635 64890d00000000  mov     dword ptr fs:[0],ecx
64a7463c 59              pop     ecx
64a7463d 5f              pop     edi
64a7463e 5e              pop     esi
64a7463f 5b              pop     ebx
64a74640 8be5            mov     esp,ebp
64a74642 5d              pop     ebp
64a74643 c20400          ret     4                  //参数个数为1
```

参数内容为0x2

```
64a74643 c20400          ret     4
0:000:x86> r
eax=64a73f10 ebx=00000000 ecx=153bb918 edx=64dd0190 esi=092c3630 edi=091f3360
eip=64a73f10 esp=04afb788 ebp=04afe750 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00200246
ConvertToPDF_x86!CreateFXURLToHtml+0x450:
64a73f10 55              push    ebp
0:000:x86> dd esp
04afb788  0159543f 00000002 8ca133eb 09b8fe56
04afb798  00000000 00000000 00000000 00000000
04afb7a8  ffffffff 00010000 0000ffff 00000000
04afb7b8  ffffffff 00010000 0000ffff 00000000
04afb7c8  04afb800 778fca3c 04afbc80 46454447
04afb7d8  09b8fa68 000003ee 00000000 04afb8d4
04afb7e8  000003ee 778fca5b 04afbca0 778fcf90
04afb7f8  0bad023c 04afbad4 04afbae0 04afbae0
```

据此可以确定另外两个函数的参数个数和内容。

网上参考的的wrapper的具体代码内容如下

```
/*
foxit-fuzz.cpp - simple console wrapper for ConvertToPDF_x86.dll
@richinseattle / rjohnson@moflow.org

NOTES:

Must install the foxit pdf printer globally
Harness targets foxit 9.0 API by default
Can target 7.3.4 if FOXIT_734 is defined and ConvertToPDF_x86.734.dll is in path

afl-fuzz.exe -i %INPUT_DIR% -o foxit_out -D %DynamoRIO_ROOT%\bin32 -t 20000 -- -
coverage_module ConvertToPDF.dll -target_module foxit-fuzz.exe -target_method
convert_to_pdf -nargs 2 -fuzz_iterations 5000  -- %CD%\foxit-fuzz.exe @@ NUL
*/

#include <Windows.h>
#include <String.h>
#include <iostream>
using namespace std;

typedef void * (__stdcall *CreateFXPDFConvertor_t)();
typedef int(__thiscall *InitLocale_t)(void *_this, int, wchar_t * lc_str);
typedef int(__thiscall *InitPrinter_t)(void* _this, wchar_t *printer_name);
typedef int(__thiscall *InitPdfConverter_t)(void* _this, int mode);
#ifdef FOXIT_734
typedef int(__thiscall *ConvertToPdf_t)(void* _this, wchar_t *convert_buf, int p2, int
p3);
#else
typedef int(__thiscall *ConvertToPdf_t)(void* _this, wchar_t *convert_buf, int p2, int
p3, int p4, int p5, int p6, int p7, int p8);
#endif

typedef struct ConverterFuncTable_t
{
    ConvertToPdf_t     ConvertToPdf;
    InitPdfConverter_t InitPdfConverter;
    InitPrinter_t      InitPrinter;
    //InitLocale_t       InitLocale;

} ConverterFuncTable;

typedef struct ConverterClass_t
{
    ConverterFuncTable_t *vfp_table;
} ConverterClass;

#ifdef FOXIT_734
char *target_library = "ConvertToPDF_x86.734.dll";
#else
char *target_library = "ConvertToPDF_x86.dll";
#endif
char *target_function = "CreateFXPDFConvertor";
```

```
wchar_t * printer_name = L"Foxit Reader PDF Printer";

ConverterClass *pdfconverter = NULL;


int init_target_library()
{
    int retVal = 0;

    CreateFXPDFConvertor_t CreateFXPDFConvertor =
(CreateFXPDFConvertor_t)GetProcAddress(LoadLibraryA(target_library), target_function);

    // create an instance of CreateFXPDFConvertor
    pdfconverter = (ConverterClass *)CreateFXPDFConvertor();
    ConverterFuncTable *vfp_table = pdfconverter->vfp_table;

    cout << "Function table:   " << endl;
    cout << "CreateFXPDFConvertor: " << hex << CreateFXPDFConvertor << endl;
    cout << "InitPdfConverter:     " << hex << vfp_table->InitPdfConverter << "
CreateFXPDFConvertor+0x" << hex << (unsigned long)vfp_table->InitPdfConverter -
(unsigned long)CreateFXPDFConvertor << endl;
    cout << "InitPrinter:          " << hex << vfp_table->InitPrinter << "
CreateFXPDFConvertor+0x" << hex << (unsigned long)vfp_table->InitPrinter - (unsigned
long)CreateFXPDFConvertor << endl;
    cout << "ConvertToPdf:         " << hex << vfp_table->ConvertToPdf << "
CreateFXPDFConvertor+0x" << hex << (unsigned long)vfp_table->ConvertToPdf - (unsigned
long)CreateFXPDFConvertor << endl << endl;

    // init converter
    retVal = vfp_table->InitPdfConverter(pdfconverter, 2);
    if (retVal)
        cout << "Error: InitPdfConverter(): " << retVal << endl;

    // init printer device
    retVal = vfp_table->InitPrinter(pdfconverter, printer_name);
    if (retVal)
        cout << "Error: InitPrinter(): " << retVal << endl;

    return retVal;
}

extern "C" __declspec(dllexport) int wmain(int argc, wchar_t *argv[]);
extern "C" __declspec(dllexport) int convert_to_pdf(ConvertToPdf_t convert, wchar_t *
converter_buf);

int convert_to_pdf(ConvertToPdf_t convert, wchar_t * converter_buf)
{
#ifdef FOXIT_734
    return convert(pdfconverter, converter_buf, 0, 0);
#else
    return convert(pdfconverter, converter_buf, 0, 0, 0, 0, 0, 0, 0);
#endif
}
```

```
int wmain(int argc, wchar_t *argv[])
{
    int retVal = 0;

    int converter_buf_count = 0;
    int converter_buf_size = 0;
    wchar_t *converter_buf = NULL;

    wchar_t *input_path = NULL;
    wchar_t *output_path = L"nul";

#ifdef FOXIT_734
    cout << "foxit-fuzz (target v7.3.4) - rjohnson@moflow.org" << endl << endl;
#else
    cout << "foxit-fuzz (target v9.0) - rjohnson@moflow.org" << endl << endl;
#endif

    if (argc < 2)
    {
        wcout << "usage: " << argv[0] << " <input> [output]" << endl;
        return -1;
    }

    if (GetFileAttributesW(argv[1]) == -1)
    {
        cout << "error: input file path" << endl;
        return -1;
    }
    input_path = argv[1];

    if (argc == 3)
        output_path = argv[2];

    // setup buffer for converting PDF
    converter_buf_count = 0x1000;
    converter_buf_size = converter_buf_count * sizeof(wchar_t);
    converter_buf = (wchar_t *)calloc(converter_buf_count, sizeof(wchar_t));

    wcsncpy_s(converter_buf, converter_buf_count, input_path, wcslen(input_path));
    wcsncpy_s(converter_buf + (0x208 / sizeof(wchar_t)), converter_buf_count - (0x208 /
sizeof(wchar_t)), output_path, wcslen(output_path));

    // create pdfconverter class and initialize library
    if (init_target_library())
    {
        cout << "Error intializing target library" << endl;
        return -1;
    }

    // execute wrapper for fuzzing
    retVal = convert_to_pdf(pdfconverter->vfp_table->ConvertToPdf, converter_buf);
```

```
    free(converter_buf);

    if (retVal)
    {
        cout << "Error: ConvertToPdf(): " << retVal << endl;
        return -1;
    }

    return 0;
}
```

重点说明的代码为

wcsncpy_s(converter_buf + (0x208 / sizeof(wchar_t)), converter_buf_count - (0x208 / sizeof(wchar_t)), output_path, wcslen(output_path));

其中0x208表示input_path与output_path的间隔距离。

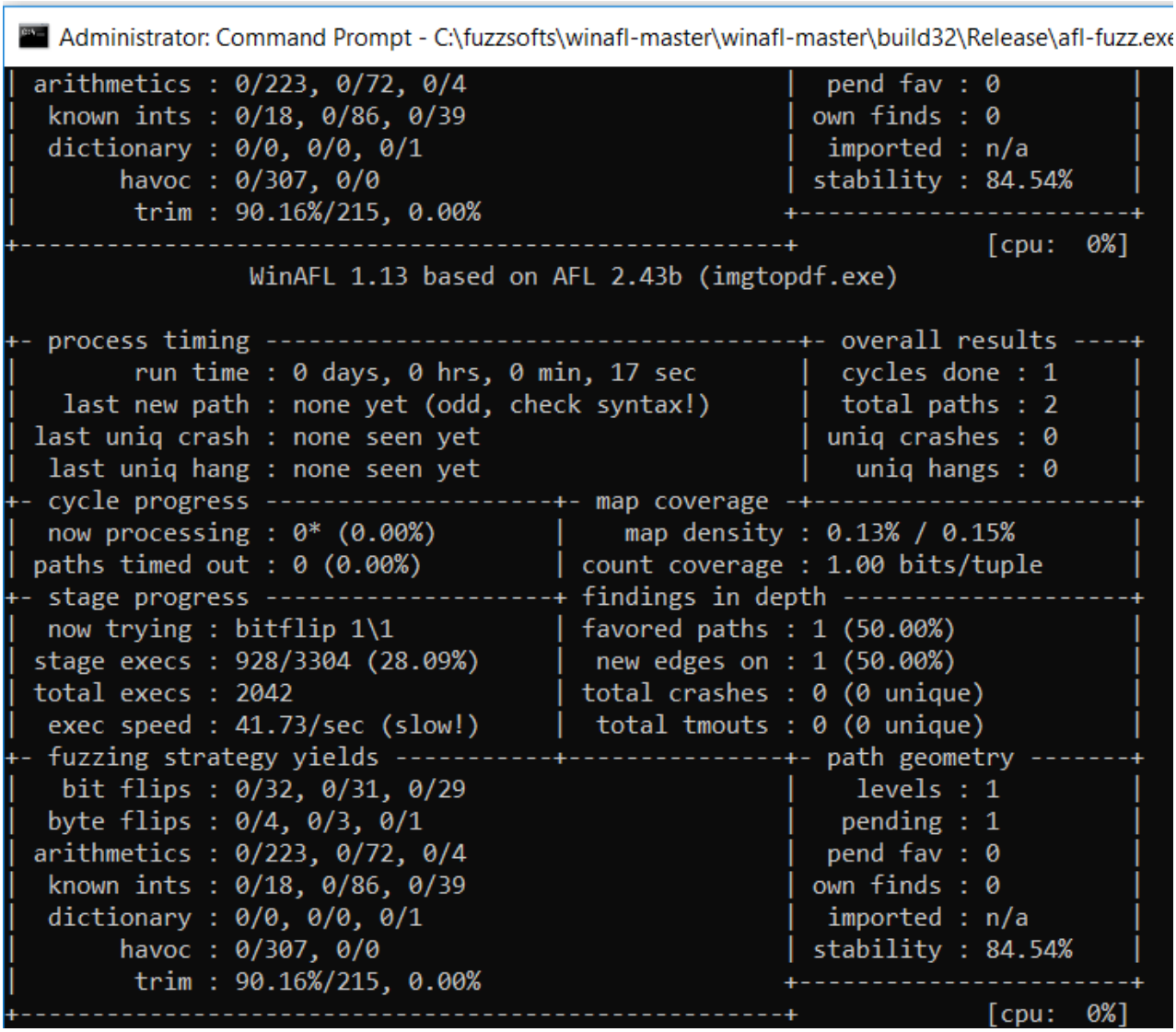
```
0:038:x86> dd esp
15c1f780  0188e7da 00cfbb54 0b9d0c20 0b9d0c20
15c1f790  00000000 15c1f7d4 00000000 00000000
15c1f7a0  00000000 7d414877 00000000 15c1f788
15c1f7b0  15c1f788 15c1f788 00000000 148db918
15c1f7c0  00cfbb54 00cfb75c 15c1f7fc 02c1a4f8
15c1f7d0  ffffffff 15c1f80c 02982989 00cfb75c
15c1f7e0  7d4147af 029829af 0b9d0c20 0b9d0c20
15c1f7f0  0b9d0c20 15c1f7e0 02997dd4 15c1f864
0:038:x86> dc 00cfbb54
00cfbb54  003a0043 0074005c 00730065 005c0074  C.:.\.t.e.s.t.\.
00cfbb64  006d0069 005c0067 002e0033 0070006a  i.m.g.\.3...j.p.
00cfbb74  00000067 00000000 00000000 00000000  g...............
00cfbb84  00000000 00000000 00000000 00000000  ................
00cfbb94  00000000 00000000 00000000 00000000  ................
00cfbba4  00000000 00000000 00000000 00000000  ................
00cfbbb4  00000000 00000000 00000000 00000000  ................
00cfbbc4  00000000 00000000 00000000 00000000  ................
0:038:x86> dc 00cfbb54 L100
00cfbb54  003a0043 0074005c 00730065 005c0074  C.:.\.t.e.s.t.\.
00cfbb64  006d0069 005c0067 002e0033 0070006a  i.m.g.\.3...j.p.
00cfbb74  00000067 00000000 00000000 00000000  g...............
00cfbb84  00000000 00000000 00000000 00000000  ................
00cfbb94  00000000 00000000 00000000 00000000  ................
00cfbba4  00000000 00000000 00000000 00000000  ................
00cfbbb4  00000000 00000000 00000000 00000000  ................
00cfbbc4  00000000 00000000 00000000 00000000  ................
00cfbbd4  00000000 00000000 00000000 00000000  ................
00cfbbe4  00000000 00000000 00000000 00000000  ................
00cfbbf4  00000000 00000000 00000000 00000000  ................
00cfbc04  00000000 00000000 00000000 00000000  ................
00cfbc14  00000000 00000000 00000000 00000000  ................
00cfbc24  00000000 00000000 00000000 00000000  ................
00cfbc34  00000000 00000000 00000000 00000000  ................
00cfbc44  00000000 00000000 00000000 00000000  ................
00cfbc54  00000000 00000000 00000000 00000000  ................
00cfbc64  00000000 00000000 00000000 00000000  ................
00cfbc74  00000000 00000000 00000000 00000000  ................
00cfbc84  00000000 00000000 00000000 00000000  ................
00cfbc94  00000000 00000000 00000000 00000000  ................
00cfbca4  00000000 00000000 00000000 00000000  ................
00cfbcb4  00000000 00000000 00000000 00000000  ................
00cfbcc4  00000000 00000000 00000000 00000000  ................
00cfbcd4  00000000 00000000 00000000 00000000  ................
00cfbce4  00000000 00000000 00000000 00000000  ................
00cfbcf4  00000000 00000000 00000000 00000000  ................
00cfbd04  00000000 00000000 00000000 00000000  ................
00cfbd14  00000000 00000000 00000000 00000000  ................
00cfbd24  00000000 00000000 00000000 00000000  ................
00cfbd34  00000000 00000000 00000000 00000000  ................
00cfbd44  00000000 00000000 00000000 00000000  ................
00cfbd54  00000000 00000000 003a0043 0055005c  .......C.:.\.U.
00cfbd64  00650073 00730072 006a005c 0068006f  s.e.r.s.\.j.o.h.
00cfbd74  0033006e 0041005c 00700070 00610044  n.3.\.A.p.p.D.a.
00cfbd84  00610074 004c005c 0063006f 006c0061  t.a.\.L.o.c.a.l.
00cfbd94  0054005c 006d0065 005c0070 00350031  \.T.e.m.p.\.1.5.
00cfbda4  00330033 00350032 00350038 00320036  3.3.2.5.8.5.6.2.
00cfbdb4  0070002e 00660064 00000000 00000000  ..p.d.f.........
```

其实不用更改上述的任何代码，直接编译即可使用

用winafl时注意命令行要改为

afl-fuzz.exe -i %INPUT_DIR% -o foxit_out -D %DynamoRIO_ROOT%\bin32 -t 20000 -- -coverage_module ConvertToPDF.dll -coverage_module foxit-fuzz.exe -target_module foxit-fuzz.exe -target_method convert_to_pdf -nargs 2 -fuzz_iterations 5000 -- %CD%\foxit-fuzz.exe @@ NUL

否则报错，运行结果如下图所示：



跑了三天，一个crash都没有，，，，，，，估计已经被很多人跑了，fuzz好难