

PFP

Python Format Parser

James Johnson

@d0c_s4vage

<https://github.com/d0c-s4vage/pfp>

What is it?

- An 010 template interpreter written in Python

The screenshot displays the 010 Editor application interface. The main window shows a hex editor with the file 'Sample.zip' open. The hex data is displayed in a grid with columns for offset (0 to 15) and rows for hex values (0000h to 00B0h). The text 'PK...UDz7.?' is visible in the hex editor. The left sidebar contains a 'Workspace' pane with 'Open Files', 'Favorite Files', 'Recent Files', and 'Bookmarked Files'. The 'Inspector' pane at the bottom left shows a table of values for the selected file.

Type	Value
Signed Byte	20
Unsigned Byte	20
Signed Short	20
Unsigned Short	20
Signed Int	20
Unsigned Int	20
Signed Int64	492384180195924...
Unsigned Int64	492384180195924...
Float	2.802597e-44
Double	1.5495355093908...

The 'Template Results - ZIPTemplate.bt' pane shows the following data:

Name	Value
struct ZIPFILERECORD record[0]	File001.txt
char frSignature[4]	PK U
ushort frVersion	20
ushort frFlags	0
enum COMPTYPE frCompression	COMP_DEFLATE
DOSTIME frFileTime	08:34:42
DOSDATE frFileDate	11/26/2007
uint frCrc	FFE73F0Ch
uint frCompressedSize	82
uint frUncompressedSize	86
ushort frFileNameLength	11
ushort frExtraFieldLength	0

The 'Floating Tab Group' window shows the 'ZIPTemplate.bt' file with the following code:

```
// Defines the end of central directory locator
typedef struct {
    char    elSignature[4];    //0x06054b50
    ushort  elDiskNumber;
    ushort  elStartDiskNumber;
    ushort  elEntriesOnDisk;
    ushort  elEntriesInDirectory;
    uint    elDirectorySize;
    uint    elDirectoryOffset;
    ushort  elCommentLength;
    if( elCommentLength > 0 )
        char    elComment[ elCommentLength ];
} ZIPENDLOCATOR;
```

What is it? (contd)

- AWESOME!



What is it? (contd)

- Quick Example (demo 1)

Motivation

- Because I can
- Lost of existing 010 templates
- Extensions
- Scriptability
- Hate being tied to one OS (e.g. windows)
- Fuzzing

How it Works

- Uses py010parser (<https://github.com/d0c-s4vage/py010parser>), a modified pycparser
- handles nodes from the parsed AST to interpret it
- Treated it ~same as browsers handle javascript (inspired at least)

Motivation

- Because I can
- Lost of existing 010 templates
- Extensions
- Scriptability
- Hate being tied to one OS (e.g. windows)
- Fuzzing

010 Compatability

- All 010 functions are (should be!) at least stubbed out
 - e.g. Strlen, FEof, FSkip, etc.
- Common ones are implemented
- SHOULD be compatible

Debugger

- 010 Editor does not let you debug your script
- Pfp does
- Demo 2

Manipulation

- 010 Editor is an *EDITOR*
- Main motivation for pfp
- Demo 3

Manipulation

- 010 Editor is an *EDITOR*
- Main motivation for pfp

```
struct {  
    int length<watch=data, update=WatchLength>;  
    char data[length];  
} main;  
  
--  
  
dom = pfp.parse(...)  
dom.main.data = "YO YO YO YO YO"  
with open("/tmp/test.bin", "w") as f:  
    f.write(dom._pfp__build())
```

Metadata

- 010 Editor metadata (aka special attributes)
- E.g.

```
struct {  
    int a;  
    int b <format=hex>;  
} main;
```

Metadata

- 010 Editor metadata (aka special attributes)
- What about length fields, checksums, etc?

Metadata - Watch

- Length fields, CRC, etc.

```
struct {  
    int length<watch=data, update=WatchLength>;  
    char data[length];  
} main;
```

Metadata - Watch

- Length fields, CRC, etc.
- Demo 4

Metadata - Pack

- Compressed/encoded data, etc

```
typedef struct {  
    int a;  
    int b;  
    int c;  
} PACKED_TYPE;  
  
struct {  
    int length;  
    char data[length]<packer=GZipper, packtype=PACKED_TYPE>;  
} main;
```


Metadata - Pack

- Compressed/encoded data, etc
- Demo 5

Vim Plugin



Vim Plugin

- <https://github.com/d0c-s4vage/pfp-vim>

EOF

