Incorrect central directory file name interpretation of Pkware ZIP file format as enabler for malicious payload delivery

Tarmo Randel
e-mail: tarmo.randel@ccdcoe.org

NATO CCD COE

7 August 2017

Abstract

The problem: miscreants are modifying ZIP file header parts so, that some automated analysis tools are unable to process the contents of packed file but targets are able to open the files with client applications and extract the possibly malicious contents. This paper points out possible negative impact of different behaviour in client implementations and proposes some solutions to address this issue.

Keywords: ZIP, malware

1 The problem

The problem occurred while processing packed file with potentially malicious content in analysis environments. The packed file in ZIP format contains Windows PE[1] file. It appears so, that the file has two names: one is with .DOC file name extension and second .exe file name extension. Detailed explanation of the problem and examples follow.

Error while processing file in the Virustotal[2]:

Figure 1. Pay attention to the SHA256 calculation line

Note, that 08.08.2017 Virustotal updated their portal engine and you will not get this picture. There are still hints that the problem has not disappeared - comparing scanning result of two identical (e.g same file is being packed) ZIP files where one ZIP is without modifications and another is modified to have different name references do provide different view of the scanning results. "Broken" ZIP will not provide "Relations" tab so probably the SHA256 hash was not successfully obtained due to error while unpacking the ZIP.

[1] Portable Executable is architecture independent executable (image) file and object file format for the Microsoft® Windows® family of operating systems, https://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx
[2] Virustotal is popular automated file and URL scanner, https://virustotal.com/

Error while unpacking file with different file name reference with Python:

```
z.extractall('tmp')
Traceback (most recent call last): File "<stdin>", line 1, in <module>
 File "/usr/lib/python2.7/zipfile.py", line 1043, in
  extract all  self.extract(zipinfo, path, pwd)
 File "/usr/lib/python2.7/zipfile.py", line 1031, in
  extract return self.\_extract\_member(member, path, pwd)
 File "/usr/lib/python2.7/zipfile.py", line 1085, in
  \_extract\_member  with self.open(member, pwd=pwd) as source, \
 File "/usr/lib/python2.7/zipfile.py", line 983, in
  open zinfo.orig\_filename,fname)
zipfile.BadZipfile: File name in directory "PO-687055.DOC" and header "PO-687055.exe" differ.
```

Opening zipfile in various environments with mismatching double file name reference - PE file has .exe listed in the local directory and .DOC in the central directory - has following results:

1. Ubuntu Ark (KDE Archiving Tool) displays error message and user can open archive file read-only. Both listed and extracted file have an .exe file name extension.
2. Unzip 6.00 by Debian and MacOS lists file name with .DOC extension[3]. Informative error message *"mismatching "local" file name, continuing with "central" file name version"* is displayed while extracting file from archive, extracted file name has .DOC extension.
3. Microsoft® Windows® 8 and 10 will display and extract file with .DOC extension. No errors or informative message is displayed.
4. MacOS user interface will display and extract file with .exe extension.
5. Ruby with zipfile[4] extracts file with .DOC extension, no errors.
6. Python throws BadZipfile exception, extracts nothing.

This issue can be re-produced easily by changing file name extension in either local, or central directory in the self-created ZIP file.

1.1 Understanding ZIP file format

In order to understand the problem we should look into raw data in the ZIP-file and available information about the ZIP-file format.

According to section 4.3.6 in [PKZIP] overall ZIP file format looks like this:

```
[local file header 1]
[encryption header 1]
[file data 1]
[data descriptor 1]
[...]
[local file header n]
[encryption header n]
[file data n]
[data descriptor n]
[archive decryption header]
[archive extra data record]
[central directory header 1]
[...]
[central directory header n]
[zip64 end of central directory record]
[zip64 end of central directory locator]
[end of central directory record]
```

Following the described format we can look up file name in two places where it is referenced: local file header and central directory header.

```
0000:0000 50 4B 03 04  14 00 00 00  08 00 C8 5D  07 4B D9 05  PK........È].KÙ.
0000:0010 A5 97 19 20  05 00 00 3C  0C 00 0D 00  00 00 50 4F  ¥.. ...<......PO
0000:0020 2D 36 38 37  30 35 35 2E  65 78 65 D4  F9 65 58 DD  -687055.exeÔùeXÝ
0000:0030 3B D3 07 8C  2E DC DD A5  B8 2C DC 5D  8A BB BB BB  ;Ó...ÜÝ¥¸,Ü].»»»
```

Figure 2. Local file name reference in the ZIP-file

Local file name "PO-687055.exe" is listed as described in section 4.3.6 [PKZIP]. Section 4.3.2 of the same document states that every local file must be accompanied by a corresponding central directory reference.

[3] unzip -l filename
[4] Ruby gem https://github.com/aussiegeek/rubyzip

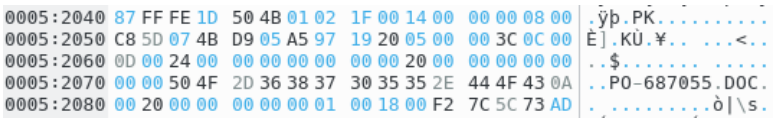

Figure 3. Central directory file name reference in the ZIP-file

We can see, that file name reference in the central directory differs from the local file name, this is acceptable in the condition described in section 4.4.17.2 [PKZIP] - file name stored in the local header will not be the actual file name if Central Directory Encryption Feature is used and general purpose bit flag 13 (see 4.3.12 [PKZIP]) is set, indicating masking.

Following described central directory format we can see, that general purpose bit flag occupies two bytes and resides in the offset 0x0009 from the central directory header. In current case the central directory section starts in the offset 0005:204c and contains:

```
central file header signature 4 bytes 0x02014b50 (constant)
version made by               2 bytes 0x001F (FAT / VFAT / FAT32 file systems, ver 3.1)
version needed to extract     2 bytes 0x0014 (2.0 see 4.4.3.2 [PKZIP])
general purpose bit flag      2 bytes 0x0000 unset
compression method            2 bytes 0x0008 (8 - The file is Deflated, see 4.4.5 [PKZIP])
```

So - the general purpose bit flag, especially bit #13 there, is not set and file name should be same for local and central directory. It is not stated explicitly but documentation defines *file name* and it is referred in both local and central directory sections.

2 Affected software, possible impact

In order to understand the possible impact and amount of affected software following activities were carried out:

1. Measuring the ability of consumer antivirus products to process ZIP files with altered file name record in central directory
2. Comparing outcome of graphical user interface and command line tools in widespread end-user operating systems
3. Mapping affected components in server side analysis environments

2.1 Antivirus products ability to cope with the situation

Testing response of end user antivirus software with both modified and unmodified ZIP files containing EICAR[5] test file lead to conclusion, that there are no critical issues with the packed file processing capabilities: AV products capable of looking into ZIP files identified the test file inside in both cases. Test results are available as Virustotal file scan results https://www.virustotal.com/\#/file/a4456e55aadaeca3fd8ef86cb477ffa38cb3ade81ceee6687f1a57c3aa4771fd (unmodified central directory file name) and https://www.virustotal.com/#/file/d77cfeb039f9ba990fd7aea57dce3d3b9b47a5b80907442ba3a994ba61399b42 (modified central directory file name).

2.2 Client applications

Client side applications for processing the ZIP files react to the described erroneous condition in two different ways. Here is the summary of this test:

Which directory used for filename	
Local	KDE (Linux) MacOS GUI
Central	Linux CLI unzip MacOS CLI unzip Microsoft Windows

[5] EICAR is organization who created anti-malware test file, more can be read here: http://www.eicar.org/86-0-Intended-use.html

Since the issue was noticed while analysing packed file with malicious content intended for targets using Microsoft Windows operating system, the file extracted has .DOC file name extension. Client application will try to open this file using client application associated with the .doc extension, which will most probably be not the desired outcome. So practical exploitation of this "feature" in the client side implementation is yet to be seen.

2.3 Affected server side components

Problems can occur with server-side software which is not prepared for previously described mismatching name condition - that could be automated malicious content analysis and detection engines and scripts. It is this type of a issue which is difficult to notice since it is not critical implementation flaw. Depending on the server side implementation described mismatching name technique can be enabler for creators of malicious files for getting desired payload through filters or content being missed for correlation.

3 How to check

Test files available for download:

- Unmodified file for reference: https://github.com/zyxtarmo/papers/raw/master/ZIPconfusion/testfiles/test_eicar.zip
- Modified file, local header lists the file inside as "eicar.com" and central directory lists file as "eicar.txt": https://github.com/zyxtarmo/papers/raw/master/ZIPconfusion/testfiles/test_eicar_broken.zip

Unmodified file should list and produce EICAR test file eicar.com.

Modified file will produce eicar.com if client application is using local header and eicar.txt if client application is using central directory header. It would be excellent if notification about abnormal condition (mismatch in filenames) will be displayed.

4 Solution

It would be helpful for developers of client implementations to find short explanation in the Pkware appnotes [PKZIP] describing how one should deal with this sort of error condition. Explanation could help to homogenize the reaction [to this erroneous condition] of the client application implementations for ZIP file format.

Most problematic (from the developers point of view) is currently Python ZIP file handling implementation. Patch for zipfile.py[6] has been prepared and pull request issued on Github, it'll take some time to spread if it will be accepted, meanwhile modified zipfile.py and patch are available for download in Github https://github.com/zyxtarmo/papers/tree/master/ZIPconfusion.

5 References

[PKZIP] PKWARE Inc.. 2014. Version: 6.3.4 .ZIP File Format Specification. [accessed 2017 August 7]. https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT

[6] created 09.08.2017