A 'Zip Bomb' to Bypass Security Controls & Sandboxes

By SANS Internet Storm Center

Archived: 2026-04-05 14:18:34 UTC

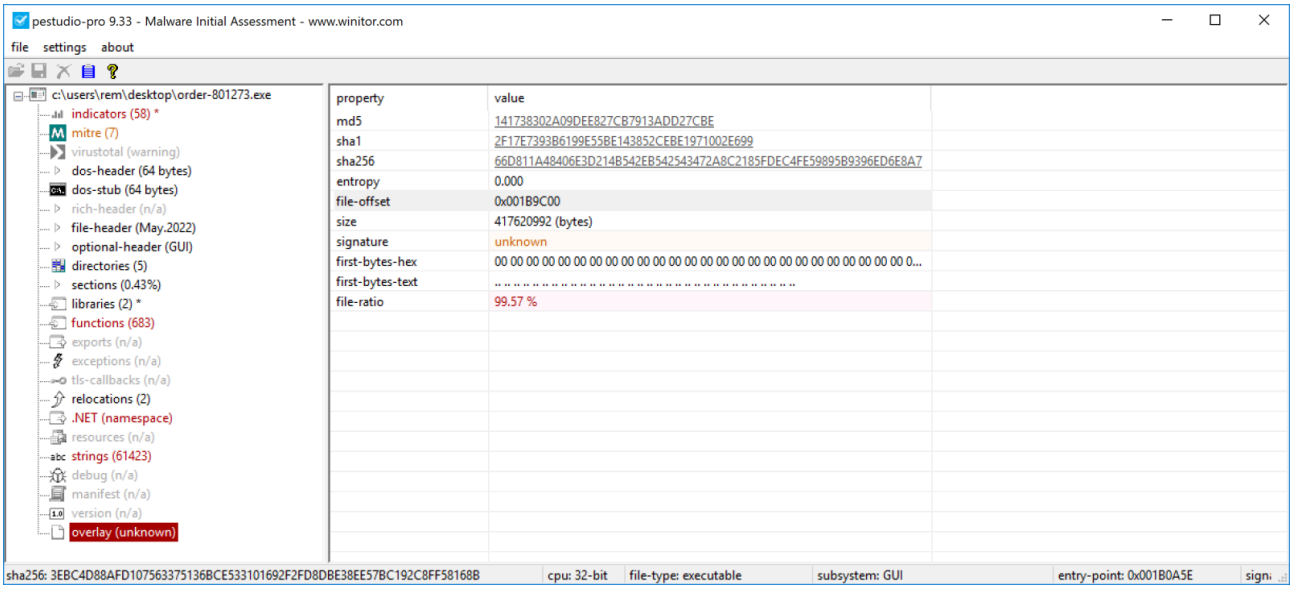
Yesterday, I analyzed a malicious archive for a customer. It was delivered to the mailbox of a user who, hopefully, was security-aware and reported it. The payload passed through the different security layers based on big players on the market!

The file is a zip archive (SHA256:97f205b8b000922006c32c9f805206c752b0a7d6280b6bcfe8b60d52f3a1bb5f) and has a score of 6/58 on VT[1]. The archive contains an ISO file that, once mounted, discloses a classic PE file. But let's have a look at the file:

```
remnux@remnux:/MalwareZoo/20220519$ zipdump.py Order-801273.zip
Index Filename         Encrypted Timestamp
    1 Order-801273.img         0 2022-05-16 13:32:08
remnux@remnux:/MalwareZoo/20220519$ zipdump.py Order-801273.zip -s 1 -d >Order-801273.img
remnux@remnux:/MalwareZoo/20220519$ file Order-801273.img
Order-801273.img: ISO 9660 CD-ROM filesystem data 'DESKTOP'
remnux@remnux:/MalwareZoo/20220519$ sudo mount -o loop Order-801273.img /mnt/iso
mount: /mnt/iso: WARNING: device write-protected, mounted read-only.
remnux@remnux:/MalwareZoo/20220519$ ls /mnt/iso
Order-801273.exe
remnux@remnux:/MalwareZoo/20220519$ cp /mnt/iso/Order-801273.exe .
remnux@remnux:/MalwareZoo/20220519$ ls -l Order*
-r-xr-xr-x 1 remnux remnux 419430400 May 20 00:34 Order-801273.exe
-rw-r--r-- 1 remnux remnux 419495936 May 20 00:30 Order-801273.img
-rw-r--r-- 1 remnux remnux   2017165 May 20 00:28 Order-801273.zip
```

Check carefully the size of the different files. The ZIP archive is 2M but the PE file is much bigger: 400MB! Do you remember the "Zip Bomb"[2]? A malicious very small archive that, once decompressed, is very big and consumes a lot of resources to be unpacked.

Let’s start the analysis of the PE file using static analysis techniques. My favorite tool to start investigations is PEstudio[3]. It reports something suspicious:



You can see (highlighted in red) that the file has an "overlay" that uses 99% of the file size! And the first bytes are all zeroes. This overlay starts at offset 0x1B9C00. Let's confirm this:

```
remnux@remnux:/MalwareZoo/20220519$ xxd -s 1809408 Order-801273.exe |more
001b9c00: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c10: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c20: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c30: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c60: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c80: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9c90: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9ca0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9cb0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9cc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9cd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9ce0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9cf0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d00: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d10: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d20: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d30: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d60: 0000 0000 0000 0000 0000 0000 0000 0000  ................
001b9d70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Microsoft Windows is very permissive regarding data appended to files. For example, it's common to see Word documents containing a macro that extracts a payload located at the end of the file. Here, the PE file has been

altered by appending a lot of zeroes to the code. That's the reason why the archive is small. Packing zeroes is very efficient and produces a small file. Let's try this:

```
remnux@remnux:/MalwareZoo/20220519$ dd if=/dev/zero of=zero.tmp count=10000000
remnux@remnux:/MalwareZoo/20220519$ zip zero.zip zero.tmp
remnux@remnux:/MalwareZoo/20220519$ ls -l zero.*
-rw-rw-r-- 1 remnux remnux 5120000000 May 19 16:06 zero.tmp
-rw-rw-r-- 1 remnux remnux    4969094 May 19 16:07 zero.zip
```

Let's get rid of the overlay to produce a new PE with a "normal" size:

```
remnux@remnux:/MalwareZoo/20220519$ dd if=Order-801273.exe of=Order-801273.exe.stripped count=180940
1809408+0 records in
1809408+0 records out
1809408 bytes (1.8 MB, 1.7 MiB) copied, 2.31218 s, 783 kB/s
remnux@remnux:/MalwareZoo/20220519$ ls -l Order-801273.exe.stripped
-rw-r--r-- 1 remnux remnux 1809408 May 20 01:01 Order-801273.exe.stripped
remnux@remnux:/MalwareZoo/20220519$ file Order-801273.exe.stripped
Order-801273.exe.stripped: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
```

Now, the file can be analyzed successfully. This is a very nice technique to bypass many security controls. Indeed, for performance reasons, big files are often skipped or can generate timeouts due to the huge amount of data to analyze.

By the way, the PE file is a bitrat sample using the following configuration:

```
{
  "family": "bitrat",
  "rule": "Bitrat",
  "c2": [
    "kot-pandora[.]duckdns[.]org:24993"
  ],
  "version": "1.38",
  "attr": {
    "tor_process": "tor",
    "communication_password": "d6723e7cd6735df68d1ce4c704c29a04"
  }
}
```

[1] https://www.virustotal.com/gui/file/97f205b8b000922006c32c9f805206c752b0a7d6280b6bcfe8b60d52f3a1bb5f
[2] https://en.wikipedia.org/wiki/Zip_bomb
[3] https://www.winitor.com

Xavier Mertens (@xme)
Xameco

Senior ISC Handler - Freelance Cyber Security Consultant
PGP Key
