SANS ISC: The DAA File Format - SANS Internet Storm Center SANS Site Network Current Site SANS Internet Storm Center Other SANS Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government OnSite Training SANS ISC InfoSec Forums

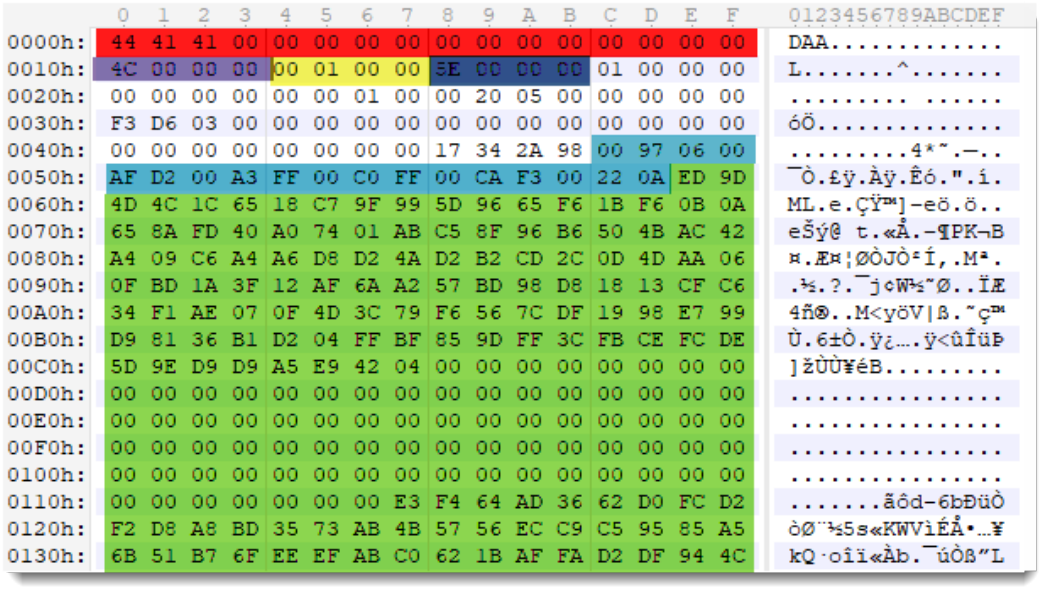
The DAA File Format

DidierStevens

640 Posts
ISC Handler
Aug 16th 2019

In diary entry "Malicious .DAA Attachments", we extracted a malicious executable from a Direct Access Archive file.

Let's take a closer look at this file format. Here is an hex/ascii dump of the beginning of the file:



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	44	41	41	00	00	00	00	00	00	00	00	00	00	00	00	00	DAA..............
0010h:	4C	00	00	00	00	01	00	00	5E	00	00	00	01	00	00	00	L.......^.......
0020h:	00	00	00	00	00	00	01	00	00	20	05	00	00	00	00	00	....
0030h:	F3	D6	03	00	00	00	00	00	00	00	00	00	00	00	00	00	óÖ..............
0040h:	00	00	00	00	00	00	00	00	17	34	2A	98	00	97	06	00	.........4*˜.—..
0050h:	AF	D2	00	A3	FF	00	C0	FF	00	CA	F3	00	22	0A	ED	9D	¯Ò.£ÿ.Àÿ.Êó."..í.
0060h:	4D	4C	1C	65	18	C7	9F	99	5D	96	65	F6	1B	F6	0B	0A	ML.e.Ç™]–eö.ö..
0070h:	65	8A	FD	40	A0	74	01	AB	C5	8F	96	B6	50	4B	AC	42	eŠý@ t.«Å.–¶PK¬B
0080h:	A4	09	C6	A4	A6	D8	D2	4A	D2	B2	CD	2C	0D	4D	AA	06	¤.Æ¤¦ØÒJÒ²Í,.Mª.
0090h:	0F	BD	1A	3F	12	AF	6A	A2	57	BD	98	D8	18	13	CF	C6	.½.?.¯j¢W½˜Ø..ÏÆ
00A0h:	34	F1	AE	07	0F	4D	3C	79	F6	56	7C	DF	19	98	E7	99	4ñ®..M<yöV|ß.˜ç™
00B0h:	D9	81	36	B1	D2	04	FF	BF	85	9D	FF	3C	FB	CE	FC	DE	Ù.6±Ò.ÿ¿….ÿ<ûÎüÞ
00C0h:	5D	9E	D9	D9	A5	E9	42	04	00	00	00	00	00	00	00	00	]žÙÙ¥éB........
00D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00F0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0100h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0110h:	00	00	00	00	00	00	00	E3	F4	64	AD	36	62	D0	FC	D2	.......ãôd-6bÐüÒ
0120h:	F2	D8	A8	BD	35	73	AB	4B	57	56	EC	C9	C5	95	85	A5	òØ¨½5s«KWVìÉÅ•…¥
0130h:	6B	51	B7	6F	EE	EF	AB	C0	62	1B	AF	FA	D2	DF	94	4C	kQ·oîï«Àb.¯úÒß”L

With the source code of DAA2ISO, I was able to make some sense of this data. I highlighted important parts:

- First we have the magic sequence: DAA...
- Second, we have an offset (0x0000004C) to the list of compressed chunk lengths
- Third, we have the file format version: 0x00000100
- Fourth, we have an offset (0x0000005E) to the first compressed chunk

- And then we have the list of chunk lengths (position 0x0000004C)
- And the chunks themselves (position 0x0000005E)

The list of compressed chunk lengths is a bit special: each lenght value is encoded with 3 bytes, using neither big-endian nor little-endian format.

The number format is the following: hex value 697 is encoded as 00 97 06. So first you have the most significant byte, then the least significant byte, and then the remaining, middle byte.

Together with the pointer to the first compressed chunk (position 0x0000005E), we can use this length list to calculate the offsets of the other compressed chunks.

Example: the second chunk is located at 0x5E + 0x697 = 0x06F3. DAA version 0x100 uses zlib compression (DEFLATE), and the compressed data is stored without header.

Armed with this information, I could write a Python script to extract and decompress the chunks stored inside a DAA file.

However, I wrote a different program. For quite some time, I was playing with the idea to write a program that can detect compressed data inside a binary stream. Since a DAA file is essentially a concatenation of zlib compressed chunks, such a program should also be able to extract and decompress the ISO file inside a DAA file.

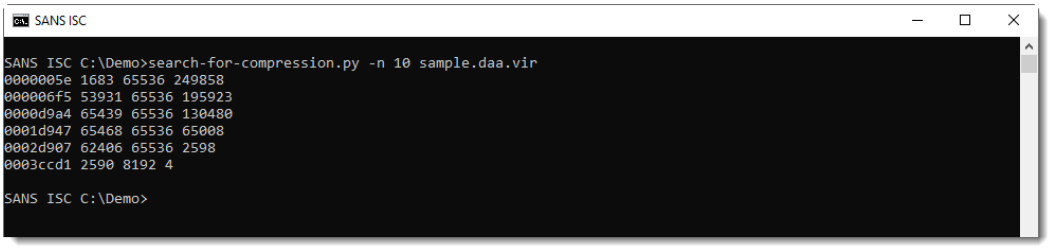
Here is the result of my beta program running on the DAA sample:

Each line represents compressed data found by the tool. The columns are:

1. start position of compressed data (hexadecimal)
2. size of the compressed data (decimal)
3. size of the decompressed data (decimal)
4. size of the remaining data (decimal)

This generic method will also generate false positives: data that decompresses but is not actual compressed data. Like the first line: it's very small (4 bytes compressed, 2 bytes decompressed) and is actually inside the DAA header. So this is clearly a false positive.

Option -n can be used to impose a minimum length on the compressed data. This can be used to filter out some false positives:



Remark that the first byte sequence of compressed data is found at position 0x5E, the same position as mentioned in the header.

And the second byte sequence of compressed data is found at position 0x6F5, that's the position that we calculated with the length of the first chunk.

All decompressed chunks have a size of 65536, except the last chunk: that's how the DAA format stores the embedded ISO file. It's chopped-up in chunks of 65536 each, that are then compressed.

Finally, I can use option -d to decompress and concatenate all compressed chunks:

A similar file format is also used for other CD/DVD image formats, like the gBurner format, compressed ISO format, ...

Didier Stevens
Senior handler
Microsoft MVP
blog.DidierStevens.com DidierStevensLabs.com