

Cracking Formbook malware: Blind deobfuscation and quick response techniques

tehtris.com/en/blog/cracking-formbook-malware-blind-deobfuscation-and-quick-response-techniques/

November 13, 2024



#Blog

#ThreatIntel

Cracking Formbook malware

Blind deobfuscation and
quick response techniques

<TEHTRIS>
FACE THE UNPREDICTABLE

Threat analysts have to respond quickly to an attack. When dealing with well-obfuscated code, shortcuts sometimes have to be taken to expedite the analysis and detection process. A malware identified by our TEHTRIS Threat Intelligence team (CTI) as Formbook stealer was writing some scrambled files on the victim's side which will illustrate how analyst could perform a known plaintext attack on an obfuscated file.

First, what is formbook ? FormBook is a widespread information-stealing malware known for targeting Windows systems. It primarily focuses on stealing login credentials, keystrokes, and other sensitive data, often delivered through phishing emails or malicious attachments. FormBook is easy to acquire as malware-as-a-service (MaaS) on underground forums, making it a popular choice for cybercriminals.

When running the FormBook malware (SHA-256: e5aebbb2c6ad445d5a2ee5c33a77d8ecfe74cf206433b723736e2e88b in our sandboxes, one file dropped by the malware (T1588.001) immediately catches our attention:

Created files (1)		
Paths	Size	SHA256
C:\Users\admin\AppData\Local\Temp\plainstones	287 KB	8101c8cc7a0668f927bb248ef147ce306f0091cfd8b05db81fbc28ba615b6c8

fig.1: Dropped file as seen in the sandbox

This file was heavily obfuscated and completely invisible in the source file. Upon inspecting it, we observed notably low entropy, which was an immediate indicator of something not encrypted. Additionally, a familiar pattern emerged, resembling structures commonly seen in a PE (Portable Executable) file. We suspect that the file is obfuscated using a static key, without the use of any diffusion mechanisms, making it easier to detect certain patterns despite the obfuscation.

The patterns, resembling the MZ header, PE header, and padding, appear to align with typical structures, as shown in the figure below.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé	Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Texte Décodé	
00000000	01	0E	75	1B	B8	51	44	31	49	14	CF	BC	39	C2	98	D2	..G..QD11.149A'O	00000000	4D	5A	50	00	03	00	00	04	00	00	00	FF	FF	00	00	MZ.....yy..		
00000010	84	0D	C2	4C	4F	95	B3	89	78	52	4C	CE	A8	DC	4C	54	...ALO*hwRLI'ULT	00000010	B8	00	00	00	00	00	00	40	00	00	00	00	00	00	008.....		
00000020	30	49	50	51	44	31	49	4C	4C	54	30	49	50	51	44	31	OIPQD1ILLTOIPQD1	00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000040	5E	4E	FE	3F	49	F8	45	99	11	F1	51	1D	89	10	1D	24	"Np?IaE".AQ.t..S	00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..*..!..Li!Th	
00000050	25	27	10	39	22	3E	23	43	28	21	6C	37	51	27	3E	3E	%'.9">#C(117Q'>>	00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno	
00000060	30	11	2B	29	6C	26	45	27	70	38	2A	11	0D	03	1F	74	0..+14E'p8*....t	00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS	
00000070	5D	22	34	34	63	3C	44	47	68	58	30	49	50	51	44	31	14444..Dp8TOIPQD1	00000070	6D	6F	64	63	2F	0D	0D	0A	24	00	0D	0D	0D	0D	0D	0D	mode.....S	
00000080	30	4D	45	F4	0D	29	37	A2	79	51	2E	BF	71	34	57	BA	QME5.)7cyQ.iq4W*	00000080	EC	BE	C4	D5	AA	DA	AA	86	AS	DA	AA	86	AS	DA	AA	86	1sAO U++ U++ U++	
00000090	4A	F7	EC	C2	73	2C	2B	A7	2A	EF	FA	A2	78	51	2E	BF	J-iAs,+5'idexQ.4	00000090	BC	B1	AB	87	AC	DA	AA	86	A1	A2	39	86	69	DA	AA	86	4ezs+U++;cs+U++	
000000A0	56	F2	9B	BA	6C	31	23	C2	1B	25	2F	3C	0D	29	37	A2	Vo**118A.w/<.)7e	000000A0	AS	DA	AB	86	98	DF	AA	86	BC	B1	AE	87	A1	DA	AA	86	Uet**s+4ezs;U++	
000000B0	44	31	49	4C	4C	54	30	49	50	14	44	31	05	4D	4D	54	DIILLTOI..DI.MMT	000000B0	BC	B1	A9	87	AB	DA	AA	86	BC	B1	AA	87	A9	DA	AA	86	4ezs+U++4ezs+U++	
000000C0	91	C6	B7	07	44	31	49	4C	4C	54	30	49	B0	51	46	30	*E..DIILLTOI*QFO	000000C0	BC	B1	A7	87	79	DA	AA	86	BC	B1	55	86	A9	DA	AA	86	4ezs+U++4ezs+U++	
000000D0	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	000000D0	BC	B1	A3	87	86	DA	AA	86	BC	B1	55	86	A9	DA	AA	86	4ezs+U++4ezs+U++	
000000E0	E0	42	44	31	49	50	4C	54	30	39	54	51	44	31	09	4C	ABD1L1TOIPQD1.L	000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000F0	4C	44	30	49	50	53	44	31	4F	4C	4C	54	30	49	50	51	LDOIPSD1OLLTOIPQ	000000F0	00	00	00	00	00	00	00	50	45	00	00	64	86	07	00PE..df..		
00000100	42	31	49	4C	4C	54	30	49	50	21	40	31	49	4E	4C	54	BIILLTOIP#@INLT	00000100	46	78	B8	EB	00	00	00	00	00	00	00	F0	00	22	20	Fx,6.....8.*		
00000110	30	49	50	51	46	31	09	CD	4C	54	20	49	50	41	44	31	OIPQF1.ILT IPAD1	00000110	0B	02	0E	14	00	02	08	00	B4	03	00	00	00	00	00	00	
00000120	49	4C	5C	54	30	59	50	51	44	31	49	4C	5C	54	30	49	ILTOYFQD1ILLTOI	00000120	10	74	01	00	10	00	00	00	00	00	80	01	00	00	00	00t.....E.....	
00000130	50	51	44	31	49	4C	4C	54	30	49	50	51	44	31	49	4C	QD1ILLTOIPQD1IL	00000130	00	10	00	00	00	02	00	0A	00	00	0A	00	00	00	00	00	
00000140	4C	54	30	49	50	51	44	31	49	4C	4C	54	30	49	50	51	LTOIPQD1ILLTOIPQ	00000140	0A	00	00	00	00	00	00	00	10	0C	00	00	04	00	00	00	
00000150	44	31	49	4C	4C	54	30	49	50	51	44	31	49	4C	4C	54	DIILLTOIPQD1ILL	00000150	1C	95	0C	00	03	00	60	41	00	00	04	00	00	00	00	00*.....A.....	
00000160	30	49	50	51	44	31	49	4C	4C	54	30	49	50	51	44	31	OIPQD1ILLTOIPQD1	00000160	00	10	00	00	00	00	00	00	00	10	00	00	00	00	00	00	00
00000170	49	4C	4C	54	30	49	50	51	44	31	49	4C	4C	54	30	49	ILLTOIPQD1ILLTOI	00000170	00	10	00	00	00	00	00	00	00	00	00	10	00	00	00	00	
00000180	50	51	44	31	49	4C	4C	54	30	49	50	51	44	31	49	4C	QD1ILLTOIPQD1IL	00000180	00	CC	09	00	58	DF	00	58	AB	0A	00	94	07	00	00	00i..Xb..Xe..	
00000190	4C	54	30	49	50	51	44	31	49	4C	4C	54	30	49	50	51	LTOIPQD1ILLTOIPQ	00000190	00	F0	0B	00	20	05	00	00	00	0B	0E	56	00	00	00	008.....E..eV..	
000001A0	44	31	49	4C	4C	54	30	49	50	51	44	31	49	4C	4C	54	DIILLTOIPQD1ILL	000001A0	00	AC	0B	00	68	41	00	00	00	0C	00	34	03	00	00	00-..ha.....4.....	
000001B0	1E	3D	35	29	30	31	49	4C	78	05	34	49	50	41	44	31	..5)0iL1x.4IPAD1	000001B0	10	AB	08	00	70	00	00	00	00	00	00	00	00	00	00	00e..p.....	
000001C0	49	1E	48	54	30	49	50	51	44	31	49	4C	4C	54	30	49	I.LTOYFQD1ILLTOI	000001C0	00	10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001D0	50	51	44	31	49	4C	4C	54	30	49	50	51	44	31	49	4C	QD1ILLTOIPQD1IL	000001D0	F0	27	08	00	18	01	00	00	00	00	00	00	00	00	00	00	008.....

fig.2: Comparison with "kernel32.dll"

We can now attempt to guess the key. By XORing the first two bytes of the obfuscated file with the known plaintext (signature of the "MZ" header), we can retrieve the first two bytes of the key. If the input buffer is zero, the key will appear in plain text. By searching for this pattern in large areas of zero bytes within the input, we can determine both the key content and its length (orange).

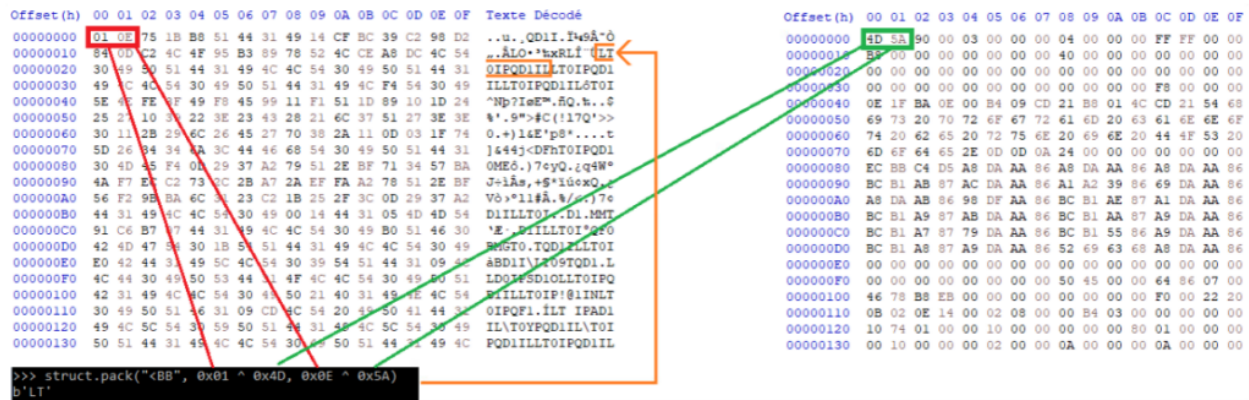


fig.3: Key guessing

A simple Python script can be used to deobfuscate this type of obfuscated file by applying the retrieved key to reverse the XOR encryption.

```
import sys
import pathlib
import array

def xor_decrypt(encrypted_file: pathlib.Path,
                decrypted_file: pathlib.Path,
                key: bytes) -> None:

    key = array.array("B", key)
    with encrypted_file.open("rb") as enc_fd, decrypted_file.open("wb") as
dec_fd:

        while chunk := enc_fd.read(len(key)):
            chunk = array.array("B", chunk)
            for i in range(len(chunk)):
                chunk[i] ^= key[i]
            dec_fd.write(chunk.tobytes())

if __name__ == "__main__":
    infile = pathlib.Path(sys.argv[1])
    outfile = pathlib.Path(sys.argv[1] + ".exe")
    xor_decrypt(infile, outfile, b"LT0IPQD1IL")
    print(outfile)
```

The deobfuscated dropped file was generated using Hasherezade. As a result, this file is already detectable by our sandbox.

[8 / 10] Yara rule detection triggered while analyzing binaries

- **Hit:** Binary triggered the Yara rule : Executable has trailing data
- **Hit:** Binary triggered the Yara rule : Inconsistant ASLR in PE header. Bad compilation
- **Hit:** Binary triggered the Yara rule : Hasherezade PE to shellcode (Peshc)

fig.4: Sandbox detection

To detect this particular type of obfuscated payload, we will apply basic mathematical checks to identify if the known plaintext pattern can be found. First, we will attempt to guess the first two bytes of the key and estimate a multiple of the key length, which should be determined quickly due to the abundance of null bytes in the PE header. Finally, we will compare two offsets that should contain null bytes in the original file (0x3FD and 0x3A0), aligned with the multiple of the guessed key length, with the first two bytes of the clear text key to confirm the key's validity.

The following YARA rule implements these calculations:

```
rule formbook_obfuscated_payload {
  meta:
    author = "PEZIER Pierre-Henri. Copyright TEHTRIS 2024"
  condition:
    filesize > 100KB and
    int8(0) != 0x4D and                                     // Regular
    PE is encrypted with null key. Blacklist it!
    for any i in (2..0x30) : (                               // Look at
the PE header only
      int8(0) ^ 0x4D == int8(i) and int8(1) ^ 0x5A == int8(i + 1) // Guess it
matches the next 2 bytes
      and (                                                  // i is a
modulo of the key length
        (                                                    // Zero
byte known plaintext attack on 2 bytes
          int8(0x3FD - 0x3FD % i) == int8(0) ^ 0x4D
          and int8(0x3FD - 0x3FD % i + 1) == int8(1) ^ 0x5A
        )
      and (                                                  // Test
another offset
          int8(0x3A0 - 0x3A0 % i) == int8(0) ^ 0x4D
          and int8(0x3A0 - 0x3A0 % i + 1) == int8(1) ^ 0x5A
        )
      )
    )
}
```

The analysis employs various techniques to expedite the process, allowing us to be more responsive to our customers. This quick “live my life” offers a brief preview of our work, which is often overlooked but crucial for maintaining security.

IOCs (indicators of compromise)

The following SHA-256 hashes correspond to similar samples:

- 08ceff2aa4f92b26b10f1accbc1d41cb2729e3beb2f558ce0fe060f77243a86c
- 0e9622e96afd3166996349ccd288105bb5c2c9c287c979ab2f0baa3b0b461929
- 10882b3477b6a32049e6f67e67885927ddcc28750884e0b02df5f228bc10f905
- 44162eee61f7d49a55fe0f815d0bc996cd728d96307b5bc6277fe430941ad068
- 47155987c94e0b921887ed3aa2278fb857781238c518fba52224728b88b0436
- 492cf9ce5a8baf6b424bf890106ba96ae824bc8ba93c4dd2da25cbf37a685c90
- 59c25af850a539e0863d6018774ac029419d1581ca8a034b1c4ce9239bd8084b
- 5e74f08923fec3a5daf99b9a6c0763b21a98226f90c537235408a4258389ca01
- 6cc54bd57057a1fc07c2726c351a42f47caef4ae05a2693fbf6b9f693c6761c6
- 7616904db54d77cb25cc58f279bdfd6ef5cbabe19573cbd781238be01daaa1c4
- 998328ecd3a13fd3287f88e37119064b3a4094d2e935786a5327d47e4ed4466b
- b3c5c896606eb408bd97f255b916cda8cf8aa4291c3f68c5108c9ff0f5b7c0b7
- b9906d121c2b4a44b38c657e3f051be5dd55fca2d8f3e51150cafad9afd77d03
- c16321285091a58a2a0e63e4d445a71d6b9a60f27a6741c0a590a4bc5290d368
- d9c7fa0a03560078e3eb0a61d42cafd68ee7c90da0ca06ca83bc05bab596f7c6
- e5aebbb2c6ad445d5a2ee5c33a77d8ecfe74cf206433b723736e2e88b9b5d78f
- e63b97535e194d90756cc01a322550d4fa41a76117799a798ea0a78c6dd940bd
- e88d0ce2a1ef65103221e16ad5a3d31c74799fa7b75dc6ccea1c2a7c8ba5a857
- f3f0ac7ba7b93d8571adfa54987fb7374451f863b44946202bc623a528fc5b5f