A Deep Dive into Packing Software CryptOne

deepinstinct.com/2021/05/26/deep-dive-packing-software-cryptone/

May 26, 2021

Learn more

May 26, 2021 | Ron Ben Yizhak

Threat actors are continuously developing and refining methods to evade detection from cybersecurity professionals. One of the more creative ways to disguise threat activity comes in the form of packing software – a technique that applies a packing algorithm on malware to

produce a file that is harder to detect, analyze, and prevent. While packers are legitimate and quite useful in helping developers protect their code against illegal copying and reverse engineering, in the wrong hands packing software can be used for more sinister purposes.

A packing software called CryptOne became popular recently among some major threat actors. It was first reported by Fox-IT that the group behind Wastedlocker has begun using it, as well as Netwalker, Gozi ISFB v3, ZLoader, and Smokeloader.

The CryptOne packer caught our attention when Emotet started using it. We followed the Emotet group closely and published multiple articles about the malware until the operation was disrupted and taken down. Some of the most recent samples that were generated before the takedown were packed by CryptOne.

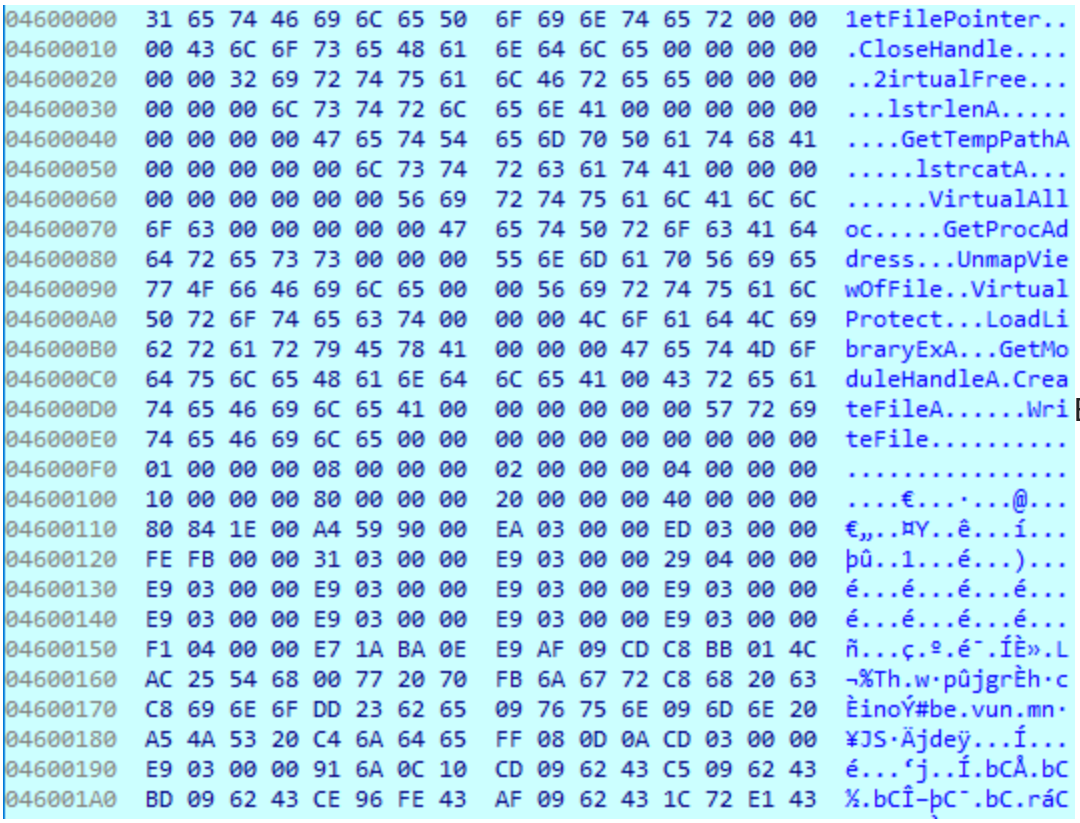
As we began analyzing the packed samples we found more malware families that are using CryptOne that weren't reported by Fox-IT, such as Dridex, Qakbot and Cobaltstrike.

In this blog post we will describe the features of this packer that made it so popular among threat actors, outline the unpacking process, and detail indicators that can determine if a sample was packed with CryptOne.

Features

1. **Multiple stages**

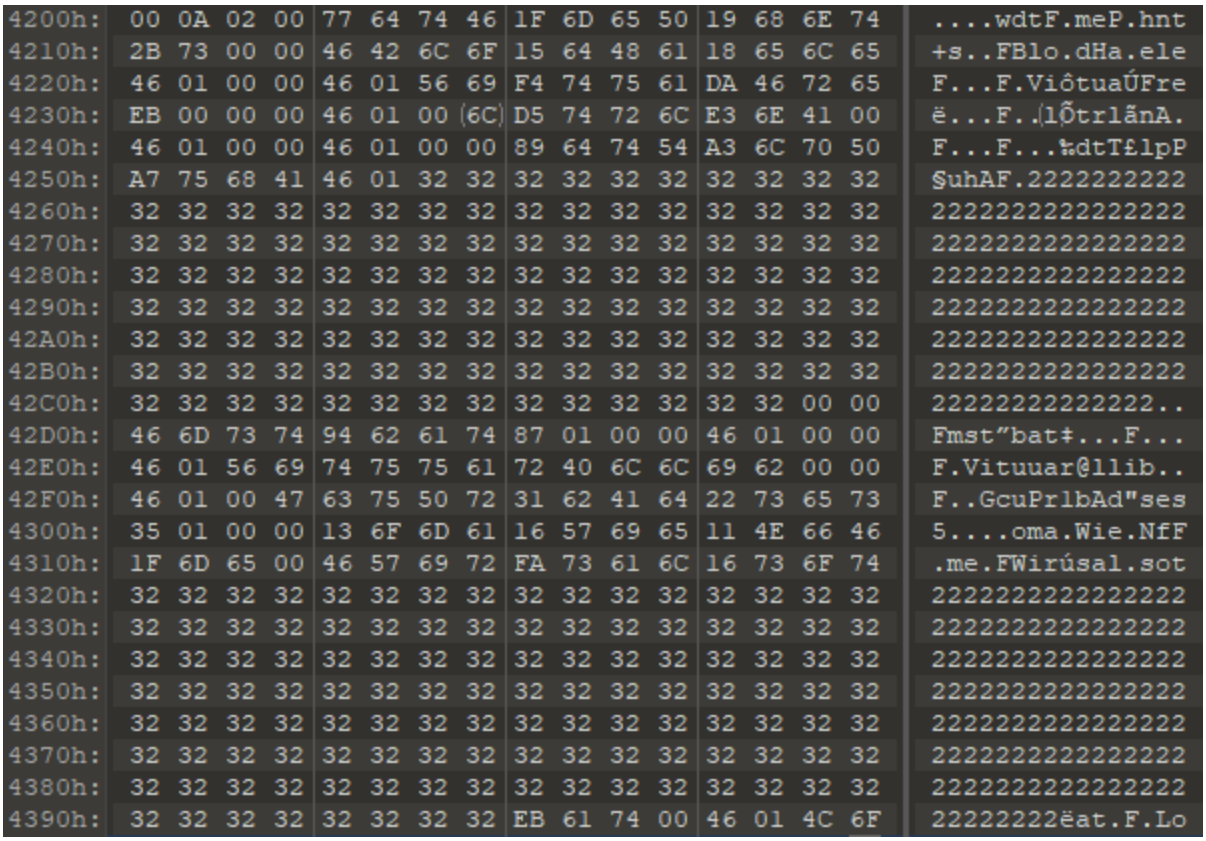
The unpacking process is composed of two stages until the destined malware is executed. The first stage is the DLL that is created by the packing software. This DLL contains encrypted data in one of its sections, which is copied to a RWX buffer and then decrypted. This data contains a shellcode and another block of encrypted data. The shellcode is described in greater detail later.



Beginning of the shellcode after the decryption

2. Reduced entropy

As mentioned, the payload is concealed as an encrypted resource. Encrypted data increase the entropy of the data and causes the loader to look more suspicious. These samples allocate a buffer with RWX permissions, but the encrypted data is not copied to it as is. Rather, it is copied in chunks while some bytes are skipped over. The bytes that are skipped over all have the same value. The reason for that might be to make the reverse engineering process more difficult, but our assumption is that the padding exists to reduce the entropy of the encrypted data and make the loader less suspicious.

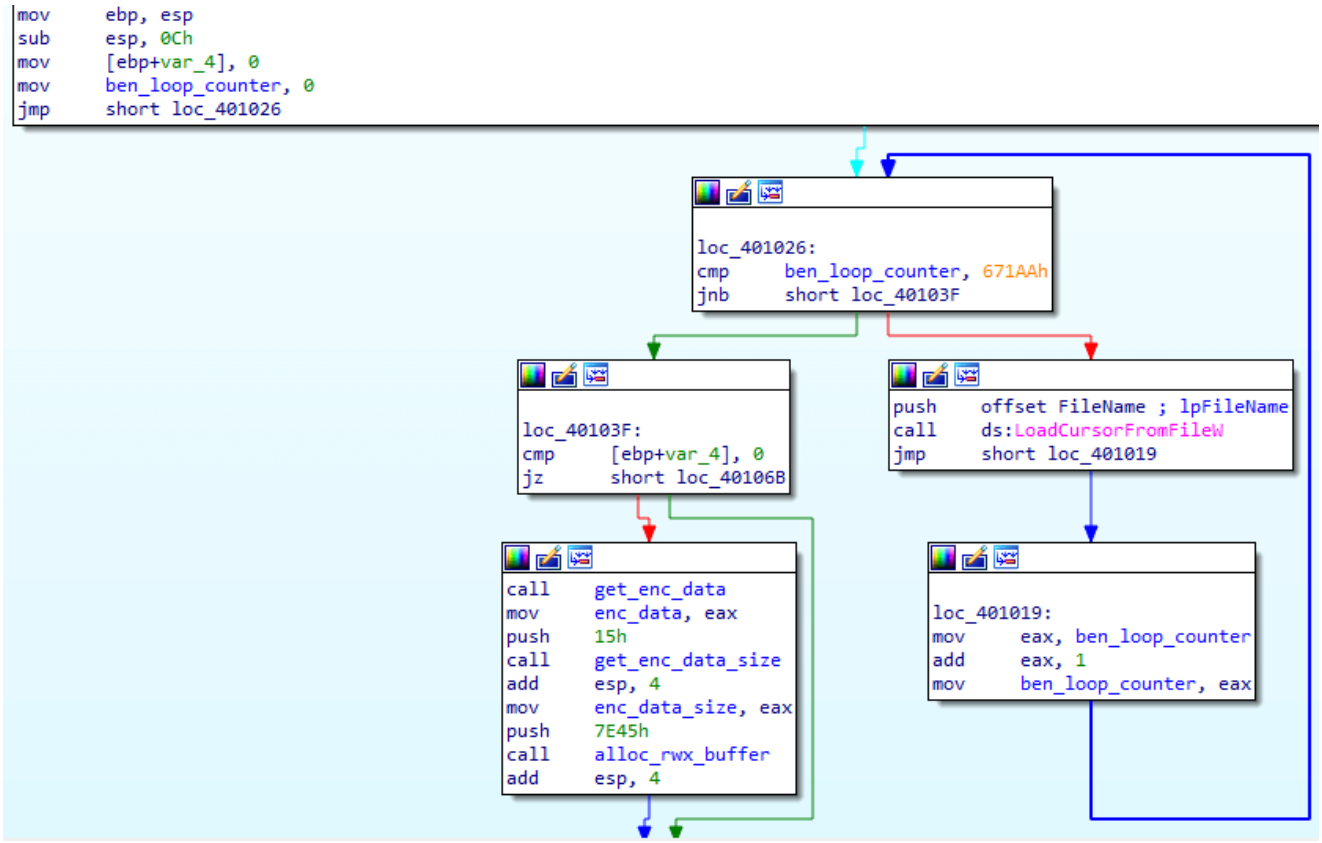


```
4200h:  00 0A 02 00 77 64 74 46 1F 6D 65 50 19 68 6E 74   ....wdtF.meP.hnt
4210h:  2B 73 00 00 46 42 6C 6F 15 64 48 61 18 65 6C 65   +s..FBlo.dHa.ele
4220h:  46 01 00 00 46 01 56 69 F4 74 75 61 DA 46 72 65   F...F.ViôtuaÚFre
4230h:  EB 00 00 00 46 01 00 6C D5 74 72 6C E3 6E 41 00   ë...F..lÕtrlãnA.
4240h:  46 01 00 00 46 01 00 00 89 64 74 54 A3 6C 70 50   F...F...‰dtT£lpP
4250h:  A7 75 68 41 46 01 32 32 32 32 32 32 32 32 32 32   §uhAF.2222222222
4260h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4270h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4280h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4290h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
42A0h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
42B0h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
42C0h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 00 00   22222222222222..
42D0h:  46 6D 73 74 94 62 61 74 87 01 00 00 46 01 00 00   Fmst”bat‡...F...
42E0h:  46 01 56 69 74 75 75 61 72 40 6C 6C 69 62 00 00   F.Vituuar@llib..
42F0h:  46 01 00 47 63 75 50 72 31 62 41 64 22 73 65 73   F..GcuPrlbAd"ses
4300h:  35 01 00 00 13 6F 6D 61 16 57 69 65 11 4E 66 46   5....oma.Wie.NfF
4310h:  1F 6D 65 00 46 57 69 72 FA 73 61 6C 16 73 6F 74   .me.FWirúsal.sot
4320h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4330h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4340h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4350h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4360h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4370h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4380h:  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32   2222222222222222
4390h:  32 32 32 32 32 32 32 32 EB 61 74 00 46 01 4C 6F   22222222ëat.F.Lo
```

Encrypted data padded with chunks of the same value.

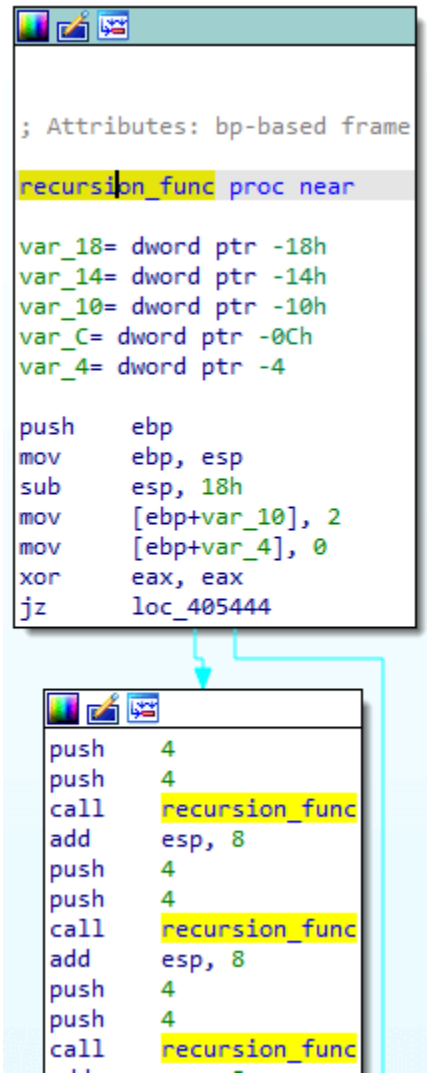
3. Sandbox evasion:

Sandboxes let the malware execute for a limited time. If the malware stays inactive until the analysis is finished, it could avoid detection. Sandbox solutions are aware of this problem, so they don’t allow the Sleep function to be used for extended periods of time.The loader created by the CryptOne software simulates Sleep. It contains small chunks of useless code that runs in loops and performs system calls that are irrelevant for the malware's functionality. In some loaders, this code executed very quickly, but the Emotet loaders took almost a minute to execute this code.Another explanation for this behavior is to fill the sandbox report with useless information so it will be harder to spot important alerts. Usually, each system call is logged by the sandbox and added to the report. The packer performs many system calls to create a report that will be difficult to process.



4. Static analysis subversion

: The loader attempts to break static analysis by inserting code blocks that will never be executed and won't interrupt the unpacking process but might confuse some disassembly algorithms. For example, a function that contains infinite recursion that will always be skipped over.

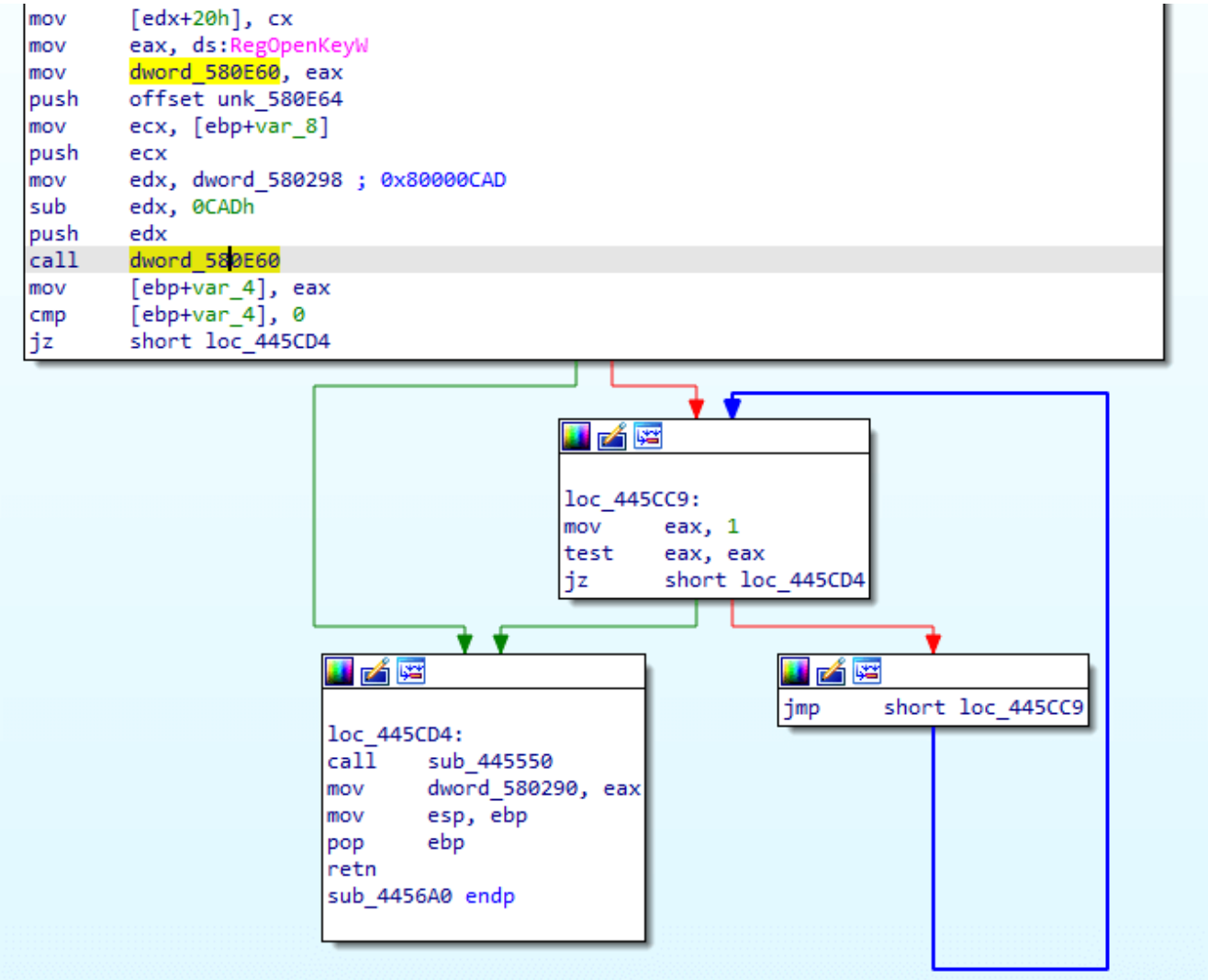


5. Killswitch

This characteristic was reported by Fox-IT

The loader checks for the existence of the registry key: *interface\{b196b287-bab4-101a-b69c-00aa00341d07}*

The loader then enters an infinite loop if the key does not exist.The loader attempts to hide the parameters that are sent to RegOpenKey. An arbitrary value is stored in a global variable. This value is then copied to a register and decreased to reach the actual value that is required for the API call. This technique was observed in multiple families. Also, in some samples the string of the registry key was decrypted in run-time.This killswitch might be a precaution to avoid infecting the control servers. Another killswitch was found only in the Emotet loaders. It exits if it is executed under the user “JhD.”



The Shellcode

The data that is decrypted by the loader has the following structure: names of WinAPI, encrypted PE file, and then the shellcode. The shellcode decrypts the PE which is the destined malware, and then performs reflective loading using the following steps:

1. Resolve the addresses of the WinAPI names. This is performed using the DLL kernelbase. This is unusual as most shellcodes use kernel32. This might be to evade detection by security products since it is known that many products place hooks inside functions from kernel32.
2. Unmap the loader image using UnmapViewOfFile. This is another uncommon technique. Usually, a new buffer will be allocated at a random address, but the shellcode of CryptOne attempts to copy the destined malware to the same address that the loader was in.
3. Copy the PE headers and sections
4. Fix the Import Address Table with the correct addresses
5. Perform the relocations listed in the relocation table
6. Change the memory protection of each section according to its characteristics

7. Update the following fields in the LDR entry of the image: entry point, DLL base, and size of image
8. Update the image base address field in the PEB structure

After all these steps are performed, the shellcode jumps to the entry point of the destined malware.

Conclusion

CryptOne is a sophisticated packer and presents a unique set of challenges to detect. It is composed of multiple stages of execution and attempts to evade detection by subverting static analysis, reducing the entropy of the data, and confusing disassembly algorithms. It also tries to avoid sandbox detection and cause damage by staying inactive for a long duration and filling the report with useless information.

These features make it attractive for attackers that need to reduce the detection rate of their malware, and we might see more threat actors use it in the near future.

If you'd like to hear more about our industry-leading approach to stopping malware, please contact us and we'll set up a demo.

/blog/why-emotets-latest-wave-is-harder-to-catch-than-ever-before

/blog/why-emotets-latest-wave-is-harder-to-catch-than-ever-before-part-2/
/blog/emotet-malware-2020/