Emotet Malware 0x01

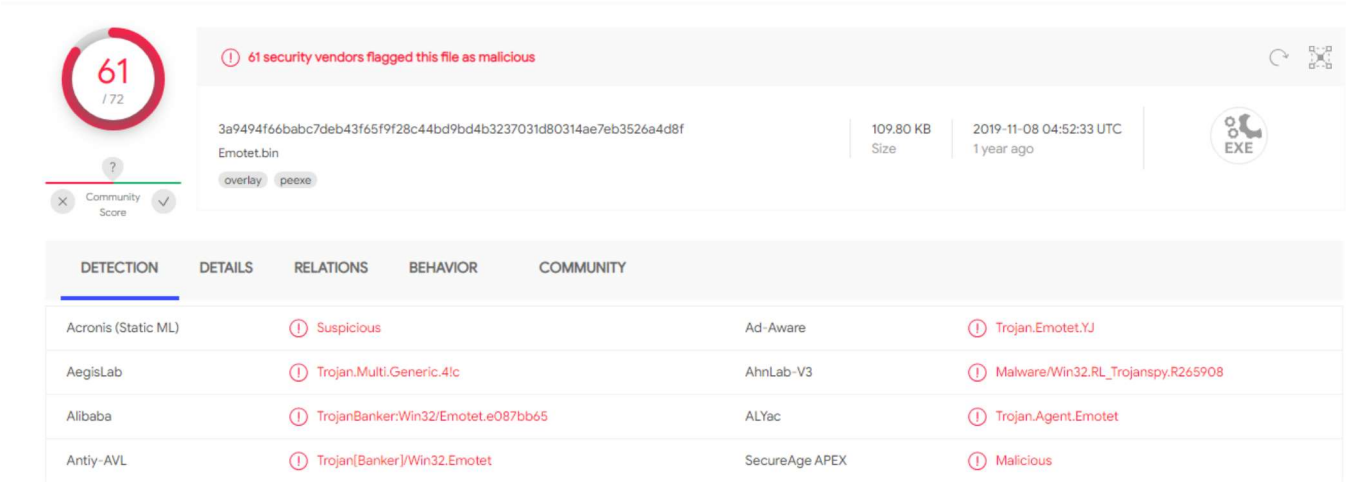
5 minute read

Introduction

The Emotet banking Trojan was first identified by security researchers in 2014. Emotet was originally designed as a banking malware that attempted to sneak onto your computer and steal sensitive and private information. Later versions of the software saw the addition of spamming and malware delivery services—including other banking Trojans.

virustotal 🔗

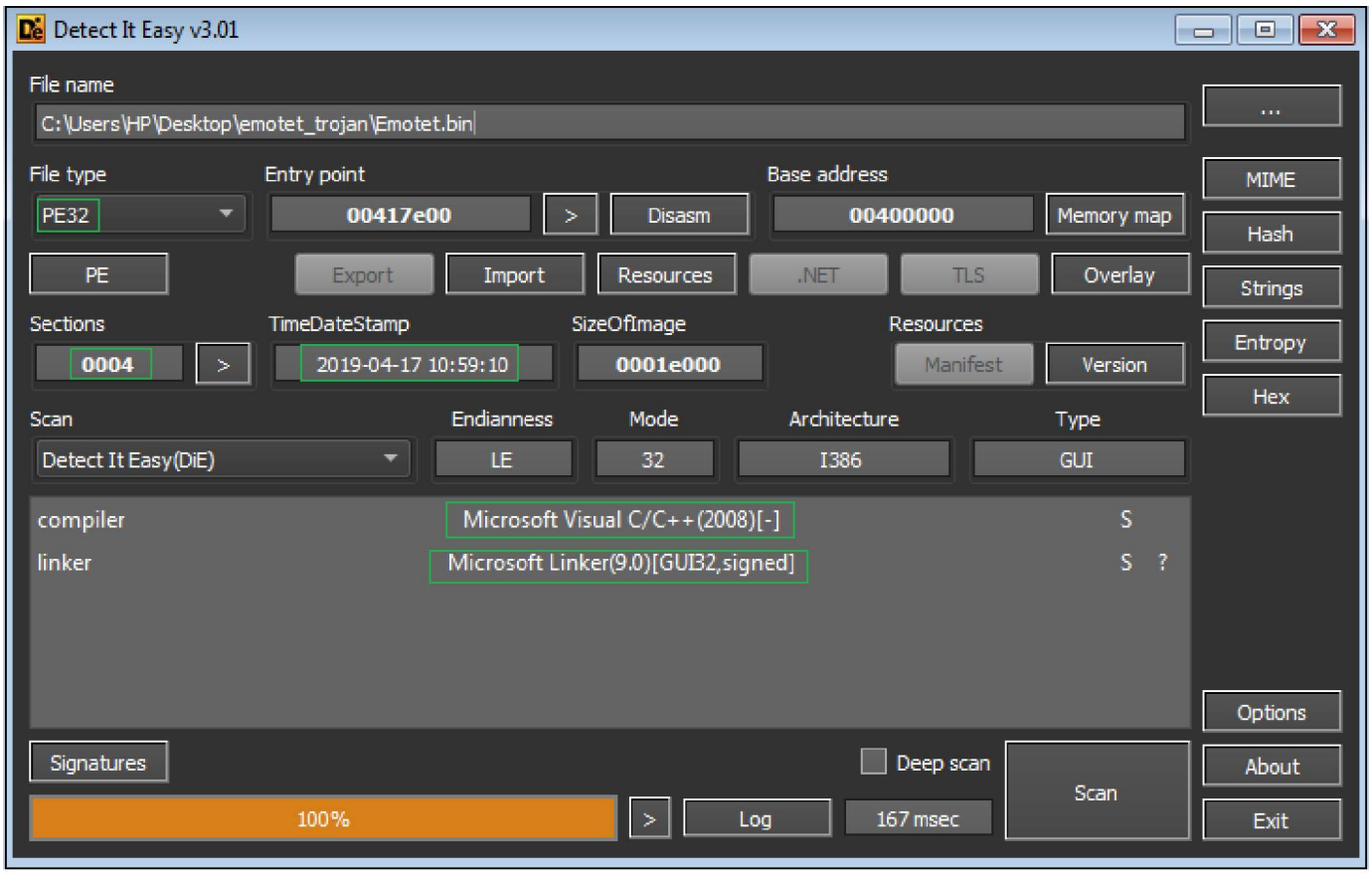
when scanning malware using VirusTotal website we can see that the malware is detected by 61 out of 70 security vendors as Emotet malware and we can see the results in the next figure.



Examine Emotet with DIE

Basic information

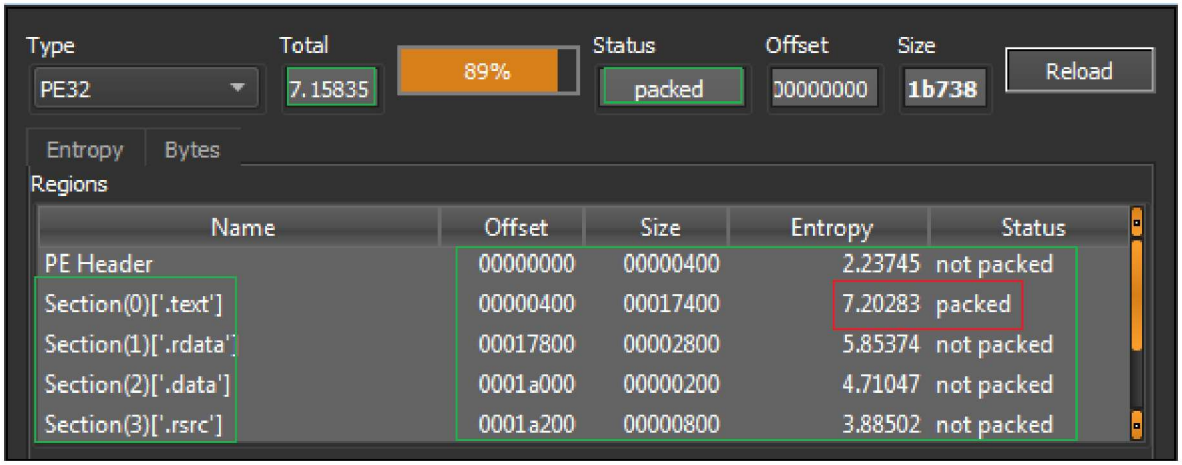
I will open malware with Detect It Easy tool to see some information about Emotet malware like compiler, type of packer (if the malware author used a commercial packer to pack it), type of linker and more information We can see the results in the next figure.



Entropy

From the previous figure, we can identify some information about Emotet malware such as:

- Compiler and linker
- The time that malware has been compiled on (the malware author can fake the time)
- The number of the sections found in the malware. Now I will examine the entropy to see whether the malware is packed or not. Entropy is a good indicator used by malware analysts to detect packing and some other techniques and we can see the results in the next figure.

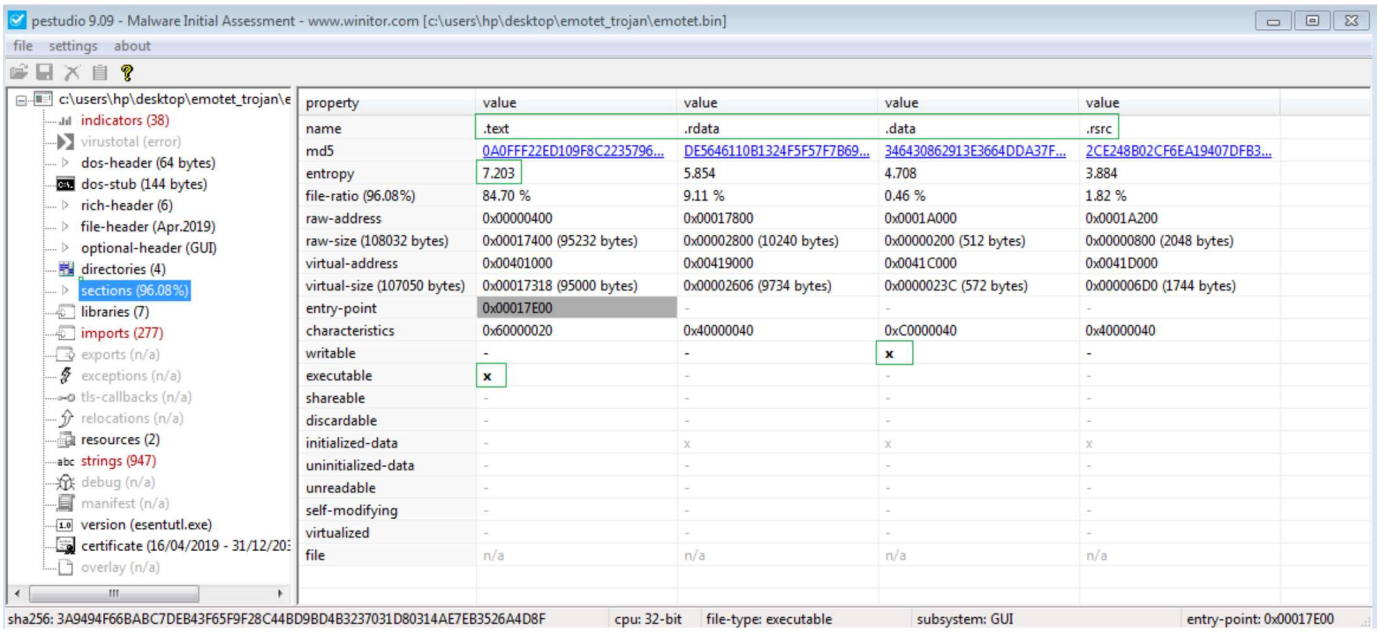


From the previous figure we can see the four sections and packed section called ".text" and see the entropy value "7.15835" which gives us a good indicator that this Emotet malware is packed.

Examine Emotet with PeStudio

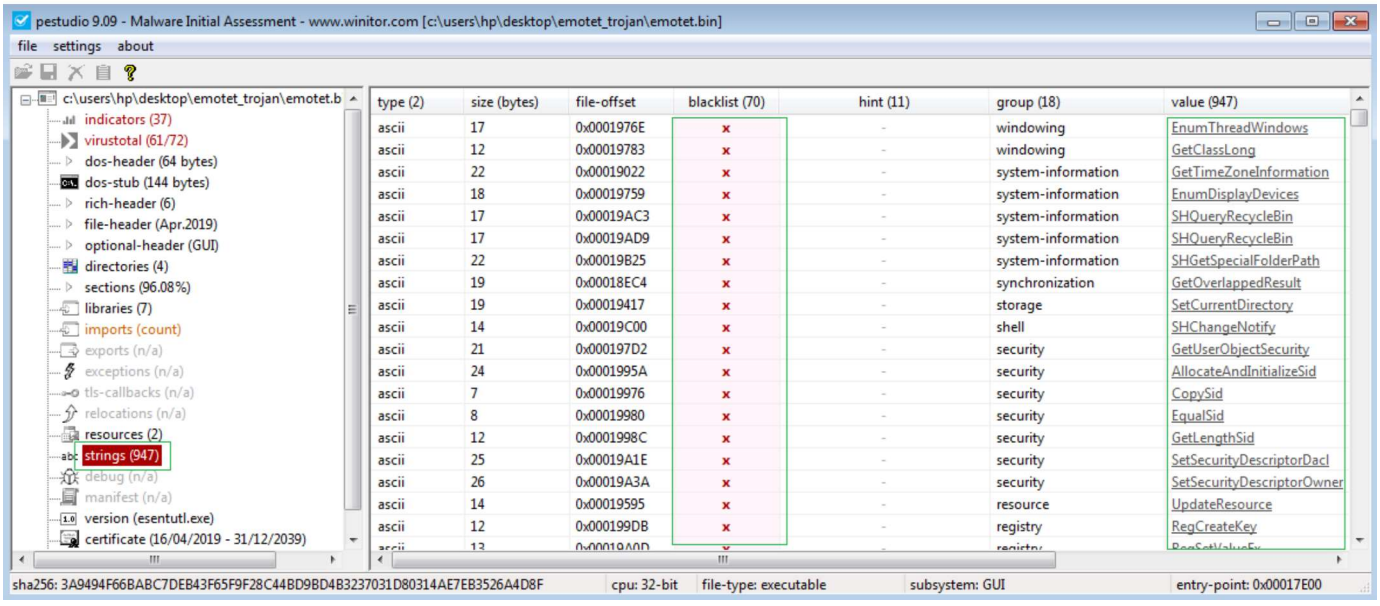
Sections

Now we will examine the section with pestudio tool to get an overview about sections and some information about every section. So, we can see four sections in the malware and we have one packed section called ".text" that has a high entropy value. We can also see that ".text" section is executable and we can see the results in the next figure.



Strings

If we examine the strings of malware, we can identify a lot of malicious strings that give us information about the behavior of the malware and what it tries to do on a system and we can see the blacklist in the next figure.

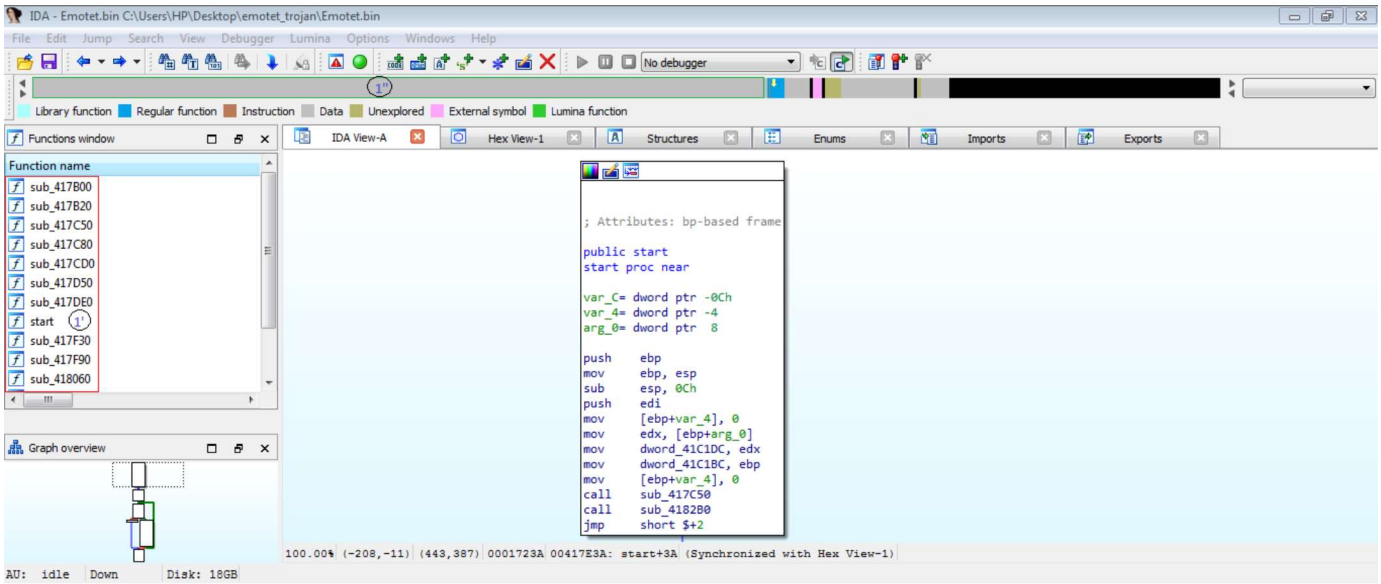


Examine Emotet with ida pro

Identify Packed Emotet

After examining the malware with a static analysis tool, we can use a great tool called "IDA Pro" to help us understand assembly code and functionalities of malware. It is also a great tool to identify packed malware with some techniques and we can see the following techniques.

- There are some of functions which used by the malware
- Most of the above bar has one color only (Navigation bar) and we can see the results in the next figure



from the previous figure, we can see two indicators telling us that this Emotet malware is packed. After examining the malware with IDA Pro, we can get also some common techniques used by malware analysts to identify packed malware.

If a malware is packed, we can search in code for:

- jump with register
- jump with a region of memory

This is because malware authors use custom packers which throw the malware into multiple loops and some encryption routines to decrypt payload in runtime (to avoid antivirus and any security solution tools). So, if we go to start function from **exports** and click on "start" function and examine the code there, we can identify an interesting function that we will deep into to see how the malware author used it to unpack the malware in runtime. We can see the name of function in the next figure from the "start" function.

Structure of function

After going into the function, we must first talk about the function's prologue and epilogue. Every function in assembly code has three parts. The first part is the "prologue", while the third part is the "epilogue". And we know that the first and third parts are simply a set of instructions that set up the context for the function when it is called or cleaned. The prologue performs such operations as:

- saving any registers that the function uses.
- allocating storage on the stack that the function needs to store the local variables.
- setting up pointers, such as parameters passed to function. The epilogue restores any saved registers, as well as the stack pointer.

syntax of function

---

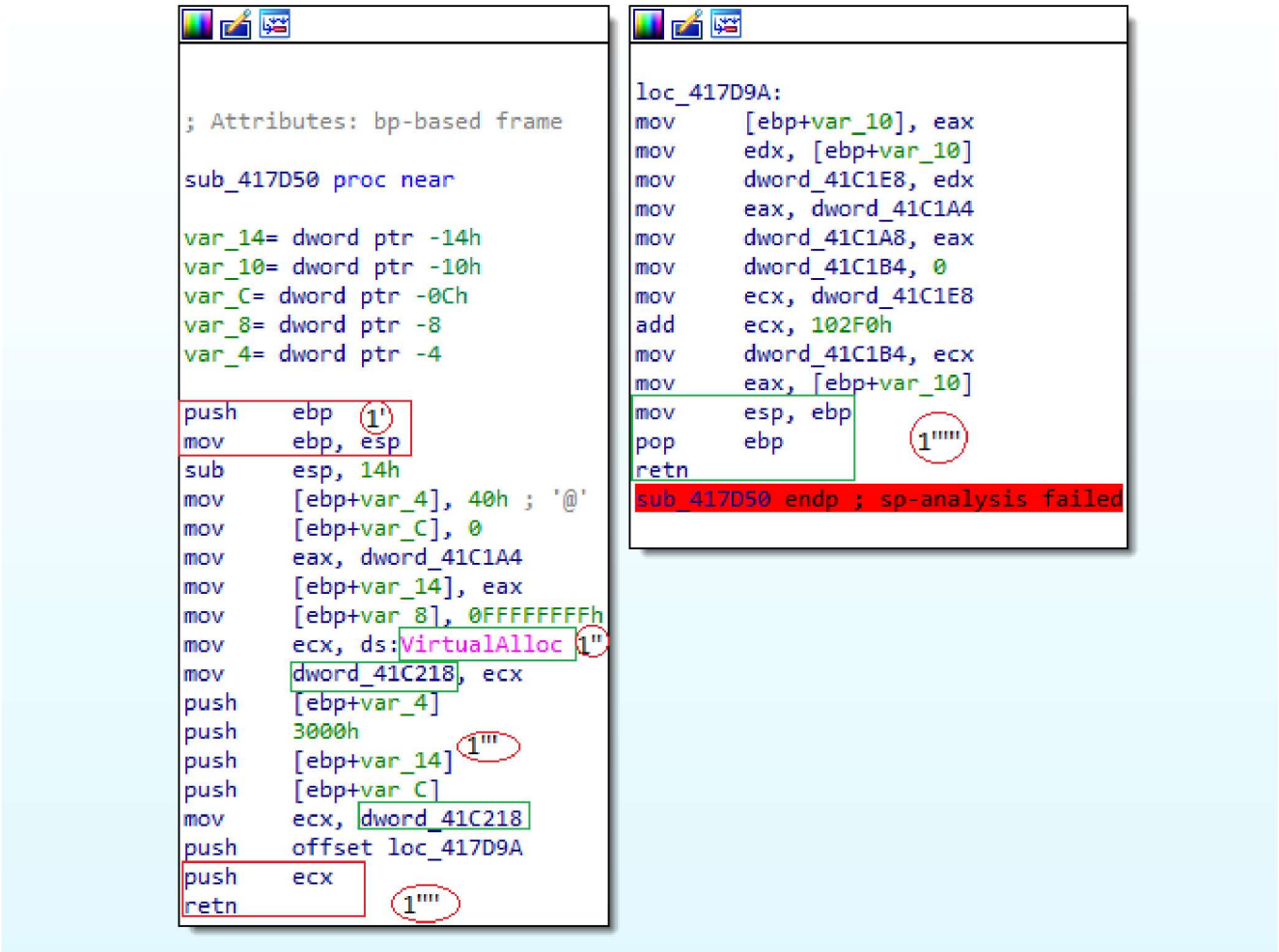
we will show the syntax in the function and this syntax depends on the architecture that we work on. So, the syntax can be different from one architecture to another.
We can see the following syntax for x86 assembly:

```
1:    push ebp
2:    mov ebp, esp    ---> prologue
3:    ...
4:
5:    main code to do operation
6:
7:    ...
8:    mov esp, ebp
9:    pop ebp          ---> epilogue
10:   retn
```

Code analysis

unpacking Function

But when we see the epilogue the function we can see that it is different from normal epilogues, and that the malware authors used it to complicate the process of reverse engineering.
We can see the results in the next figure

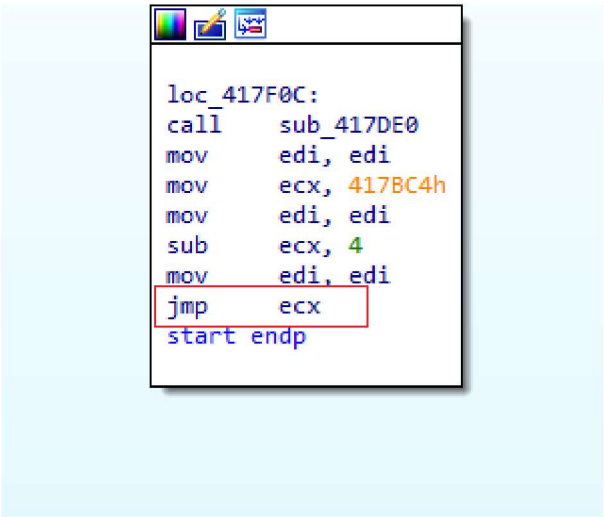


From the previous figure. We can see that:

- `VirtualAlloc` is moved into `ecx`
- Then `ecx` is moved into `dword_41C218`
- And also `dword_41C218` is moved into `ecx`
- Finally, we can see push `ecx` follwed by `ret` instruction.
  So we can identify that function is used to unpack malware.

jump Instruction

If we return to the start function and dive deep into code, we can see `jump ecx` . This is a good start for debugging the code in x32dbg.
We can see the instruction in the end of the “start” function:



We will show how to play with the code in the debugger to unpack Emotet malware in the next part.

Categories: Malware-Analysis

Updated: August 1, 2021