Recent Developments in Electronics and Communication Systems
KVS Ramachandra Murthy et al. (Eds.)

doi:10.3233/ATDE221253

Malware Reverse Engineering to Find the Malicious Activity of Emotet

Manohar Venkat G, Saranya Chandran and Arjun T U
Amrtia Vishwa Vidyapeetham, Amritapuri, Kerala, India

saranyac@am.amrita.edu

Abstract. Emotet is a Trojan that is commonly spread through emails. It was initially designed to steal banking credentials. It uses a number of strategies and infection vectors to spread over space and establish persistence on infected devices. This paper proposes a framework for analyzing Emotet malware through the process of reverse engineering, to reduce this time consumption we have researched some function calls that can help us in understanding the activity and where to locate the payload. The research is done for two types of files only, they are EXE and DLL files. Firstly we analyze the PE structure of the file using CFF explorer and check for irregularities in the address of the header. using Ghidra we further our analysis of the sample to check for irregularities, API calls, strings and many other information relating to structure of our file. On finding the common functionality and understanding its usage we can determine the kind of behavior the sample would perform and the API calls used for malicious activity. Based on the malicious activity performed we will determine whether the sample provided is Emotet or clean.

Keywords: Reverse engineer · Ghidra · Emotet malware.

1 Introduction

One of the more prevalent and effective malware suites now in use is known as Trick Bot or Emotet[1]. It uses several strategies and infection vectors to spread over space and establish persistence on infected devices [2-5]. Figure1 shows the trend of Emotet malware [6-7]

We have come up with a framework that helps in narrowing down an undeter- mined file whether it is an Emotet malware or not. Emotet is majorly distributed using emails. The logic behind the spread of Emotet malware is that it has its own list of passwords which is used as a brute force technique to gain access to different systems in a network, while the credential stealer is made up of various payloads. Our research aims to reduce the time taken to reverse engineer an Emotet malware by already knowing the type of functions that could lead to malicious activity, and the return address of the methods that will be helpful in dynamic analysis.

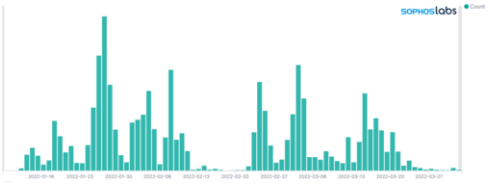

Fig. 1. Timeline of 2022 Emotet detections in SophosLabs' sandbox systems

2 Related Work

2.1 PE Structure analysis

To begin with reverse engineering, we had to understand the samples that we are dealing with and we choose to focus on PE file structure as it has more reach than being Windows executable only. The PE file structure has 5 main components like DOS header, DOS stub, PE header, section table, and n number of sections. Some of the most common section present are .text, .idata, .edata, rsrc, .rdata and .debug [8-9]. Malware Analysis In reverse engineering, we have certain steps that need to be taken.
Analyse the file using tools like PEStudio, CFF and look for strings which would be malicious, for example SQL queries, shell codes, etc. Run the file and monitor the behavior using tools like procmon, wireshark, fiddler, regshot. The result of this analysis will give out a log file showing which shows every process, registers and memory values changed on execution[10-12]. Drawbacks There are multiple drawbacks to the manual analysis performed: They are slow and time taking and in order to reduce the time consump- tion specific scripts have to be developed which is catered to the problem statement. If the file is obfuscated in such a way that they header itself is unreadable then the tools utilized to analyze the file will not work. We would have to manually build the File structure and then analyse.

2.2 Emotet and Their Method of Detection

File Analysis In the most recent years, there has been a trend upward in de- tecting Emotet malware. The file is generally spread through email. Once the attached files are downloaded into the host system, there is a high chance some diverging techniques are utilized, like the icon of the file is of PDF type but the file is of type EXE, so the user may execute the file knowing it to be a PDF file [8]. If the malware is executed it creates multiple command line instances and executes its programs. Such programs are designed to steal credentials. The command line creates a temp folder in which another binary file is dropped that does the malicious activity of stealing credentials from the cookies and sending it over network[8]. There are other ways to spread the malware that is through the brute force approach. The sample creates various ".tmp" files which look harmless when scanned. These files are also dropped in the temp folder of the windows system. On executing the dropped file, the payloads are executed which installs a password recovery tool in the system and hunts for passwords[8].

In Networks

To block the spread of Emotet malware, there many network security methods are applied. One such method is a combination of a graphical network security model and a game theory model to detect and remove the malware[5]. There are ways that Emotet malware can avoid Machine Learning model detection by moving the binary section or the payload to a resource section [4].
Using Machine Learning: There are ways to detect Emotet malware using machine learning and their data helps us in analyzing the API sequence utilized. Using the word2vec model a detection model was made for Emotet [6]. Another method for detecting Emotet is understanding the control flow graph. based on the various control flow graphs, we can find some similarities in the same set of Emotet files [3][12].

3 Methodology

3.1 Brief on Analysis of Our Sample

To analyse our sample we carry out certain steps.

- Gathering of file information.
- Understanding PE structure.
- Finding function calls.
- Finding addresses of the return function from the methods present.
- Utilise the addresses to create break point in dynamic analysis

3.2 Detail Analysis of Our Sample

This research deals with EXE and DLL files only. We have analyzed a total of 53 malware samples taken from malwarebazaar.com. The validity of our sample taken are already determined by another malware researcher and marked as an Emotet which they have published in the malwarebazaar.com website. We started our analysis by taking one of the samples and understanding the PE structure of our sample. To begin with our analysis we study the structure of our sample. To do this we analyse it using CFF explorer. CFF explorer is a tool that displays the complete PE file structure only, non PE files will not work. In our analysis we we focus on the 6 sections export, import, resource, exception, relocation, and TLS directory, figure3.They are defined as The export directory also written as ”.edata” contains names and addresses of the exported functions. The import data also written as ”.idata” contains names and addresses of the imported functions.
Resource section (.rsrc) contains information utilised by like, the information is like icon, icon size and sometimes imported functions too. The exception section holds the addresses of the exceptions utilised. Relocation sections contains addresses for the unallocated directories. Thread Local Storage (TLS) stores addresses and these addresses are first executed before the execution of the magic bytes in the PE file.

The export section is of main importance to us. So we dig deeper into the export directory and see that there are many methods/functions present, table1. We filter out some of the well-known functions of Emotet malware and understand the work behind the malware, table1. The export directory tells us different functions utilized/written by the malware author on how the flow of execution should take place . By analyzing the functions we found that the sample has a memory allocation in which a new binary file is being dropped. This dropping of binary files is a technique used by Emotet malware to download/ create another malicious file that possibly infects the system. On analysis of multiple samples we were able to narrow down to some malicious activity using the functions that could be used to run the malware activity. We used the Ghidra scripting to fetch Reference Table object and Symbol Table object to map out the functions. On understanding the function activity we can narrow down activity of the file like memory allocation functions are used for dumping files in the %TEMP% folder, such similar functions are recorded.Next we needed to get the return address of the function to keep a break point in our sample analysis. the reason behind keeping a break point is during dynamic analysis we need to figure out at what point binary file is being created. The binary file does the malicious activity of Emotet and it is also drops multiple binary files[10]. To find this break point address in the quickest way possible, we proposed an algorithm.

```
Current program
Address current program
R: Reference table
S: Symbol table
I:  Get instruction from X
P: Get process information from X
F: From address
T: To address
Method Function Node (F)
   while X = true do
            O = P
      while P = true do
            F = Address (P. from)
            T = Address(P.to)
       while T! = null do
            F = R (T)
        end while
           print (S (F))
        Function Node (F)
       end while
    end while
```

The algorithm returns the address of the function and its calling functions. So every recursive call made, will check for the function's address and based on the string given" func", it will check for the next function call jump made inside the function. This loop keeps repeating until the string obtained is null. Once null is obtained it returns the value and prints the addresses. These addresses are important as we will utilise it in dynamic analysis. The addresses displayed are in relative virtual addresses (RVA) format. We

need to convert it back into virtual address to jump to a memory location in x32/64dbg tool. The equation is
RV A = V A − ImageBase

Dynamic analysis

We selected the Virtual Alloc function to look for the address break point. VirtualAlloc is used for allocating memory and by this we can guess that VirtualAlloc is used for dropping binary file. The address is ”040030D8”, figure2. We enable a break point using our x32dbg tool4.

From the x32dbg using scyllahide plugin we dump our file, figure4. By comparing the functions and dropping of a binary file on analysis which shows password stealing techniques like SQL injection, shell code, etc during behavior analysis are Emotet.

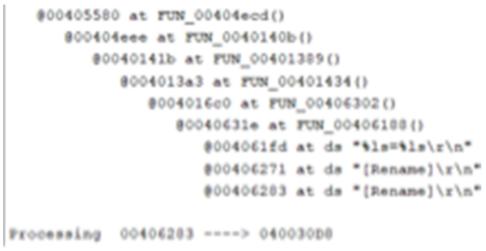

Fig. 2. Addresses retrieval from VirtualAlloc function

Fig. 3. CFF explorer view

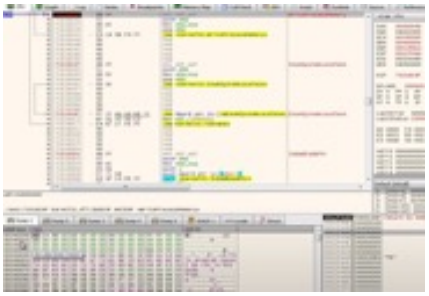

Fig. 4. Dropping malicious binary

4 Results and Discussion

4.1 Function Calls Found

The results obtained are list of functions that are used as malicious behavior in our sample and addresses related to the malicious activity.

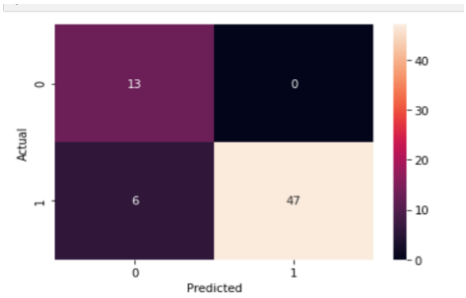

Fig. 5. A part of the data set of functions used for malicious activity

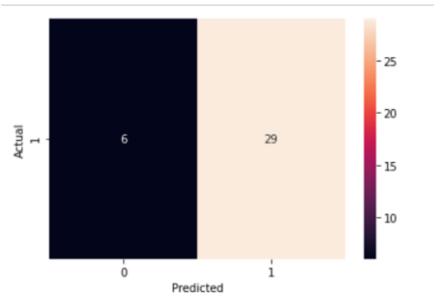

Fig. 6. Tested on undetermined samples.

4.2 Analysis of Multiple Sample

The data set is prepared for the functions found with the addresses for 53 mal- ware samples, table1. The malware samples are obtained from malware- baazar [2]. We have also tested with 13 legal files taken from the system32 folder in Microsoft Windows 11. To determine whether this malware was Emotet or not, certain key factors had to be observed, like memory allocation, process creation, string operations, exception handling, and privilege (priority to process) functions, later for dynamic analysis, we forcefully dump hidden files and per- form string analysis as we already know malicious sequence [1]. The framework we built for the analysis of Emotet malware works up to 90.9% accuracy. Out of 66 samples, 60 samples were correctly determined and 6 samples we were in- correct in determining because the address pointed was incorrect and also lack of knowledge in reverse engineering figure5. In the 53 Emotet samples taken, 18 were manually analyzed and considered to be genuinely malware,, whereas 35 samples were tested, and out of this 35 samples, 6 samples were unable to be determine, figure6. The accuracy for detecting Emotet in Emotet cases is 82.85

5 Conclusion

After testing for 53 malware samples and 13 legal Microsoft samples taken from the system32 folder, we found that the functions obtained and the addresses to the functions that we considered are being used for malicious activity were effective in analysis by shortening our manual time consumption for reverse engineering and analysis of the sample. This framework will help in determining undetermined Emotet files. For future work, we can extend the project to look for other malware like Ransomware, Worms, Trojans, bots, etc. We can also enhance the code to look for broken PE header or cor- rupted PE files.

References

[1] Malapi.io, https://malapi.io/
[2] Malware bazaar, https://bazaar.abuse.ch/browse/
[3] Anju, S., Harmya, P., Jagadeesh, N., Darsana, R.: Malware detection using assem- bly code and control flow graph optimization. In: Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India, pp. 1–4 (2010)

[4] Ceschin, F., Botacin, M., Gomes, H.M., Oliveira, L.S., Grégio, A.: Shallow se- curity: On the creation of adversarial variants to evade machine learning-based malware detectors. In: Proceedings of the 3rd Reversing and Offensive-oriented Trends Symposium. pp. 1–9 (2019)
[5] Grammatikakis, K.P., Koufos, I., Kolokotronis, N., Vassilakis, C., Shiaeles, S.: Understanding and mitigating banking trojans: From zeus to emotet. In: 2021 IEEE International Conference on Cyber Security and Resilience (CSR). pp. 121–128. IEEE (2021)
[6] He, Y., Geng, S., Zhao, Y., Feng, Y.: Research on intelligent detection method of malicious behavior based on self-attention. In: ICMLCA 2021; 2nd International Conference on Machine Learning and Computer Application. pp. 1–5. VDE (2021)
[7] Monnappa, K.: Learning Malware Analysis: Explore the concepts, tools, and tech- niques to analyze and investigate Windows malware. Packt Publishing Ltd (2018)
[8] Nagy, L.: Exploring emotet, an elaborate everyday enigma. Virus Bulletin (2019)
[9] Nisi, D., Graziano, M., Fratantonio, Y., Balzarotti, D.: Lost in the loader: The many faces of the windows pe file format. In: 24th International Symposium on Research in Attacks, Intrusions and Defenses. pp. 177–192 (2021)
[10] Sukul, M., Lakshmanan, S.A., Gowtham, R.: Automated dynamic detection of ransomware using augmented bootstrapping. In: 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI). pp. 787–794. IEEE (2022)
[11] Team, S.R., et al.: Emotet exposed: looking inside highly destructive malware. Network Security **2019**(6), 6–11 (2019)
[12] Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., Pham, T.D.: A two-stage deep learning framework for image-based android malware detection and variant classi- fication. Computational Intelligence (2022)