CHAPTER I
INTRODUCTION

1.1 PROJECT DEFINITION

In past, it is believed that steganography techniques were first used in Greece during its Golden Age, used to hide information in carrier in ways that prevent the detection of hidden message by an attacker. It is used during World War I in order to establish a secure communication path without consciousness of third-party. The attacker should not get any of the information by simply looking at the stego-text (P. Bateman, 2008).

However, the trend is change today where the attackers manipulate the technique of steganography by injecting the malware code inside a legitimate file and release it through the Internet. Basically, malware that is implanted in the files is designed not to be disruptive or intrusive but rather to stay dormant until it carrying out its malicious activities.

After the infected file being downloaded by certain user, the attacker later at a suitable time will take action by injecting the exploit into the computer, and then triggers will search the embedded malware in the computer to activate the code. It will later infiltrate subsequently depending on their objective. Besides, the embedded malware is hard to be discovered by signature-based antivirus detector even if a malware's exact signature is present in the detector database (Shafiq, M., S. Khayam and M. Farooq 2008).

In this research, our scope is to investigate the presence of malware in JPEG file. To achieve this, we infect malware JPEG files and implement this experiment where JPEG file and malware files are grouped into K=10 and use statistical profiling measures to check the binary perturbation in the n-grams of the infected JPEG files. We also want to explore the capability of our detection system towards recognising the approximate location of the embedded malware.

1.2 PROBLEM STATEMENT

Nowadays, computer users download contents such as pictures, music, document and multimedia content from the Internet without having concern if those files might contain malware code that is embedded together or not. The hidden code can be activated by the attacker and damage theirs system. Our scope of research is restricted on JPEG images as it not only come from the malicious website but it also can obtain from a legitimate website which is cracked by attacker. Thus, a solution that suits today's technology is really needed for detection and prevention for this kind of threat.

1.3 PROJECT MOTIVATION

For a threat that remain undiscovered in wild for a long time may give tons of opportunity for it to compromise computers before any countermeasures is taken to protect against it (Shafiq, Khayam and Farooq 2008). The serious implication of the malware affected the computer system today motivate us to investigate this problem. We hope this new research will help malware research community towards improving the potential of effective malware detection systems.

1.4 RESEARCH OBJECTIVES

The objectives of this research are:

1. To survey the existing research in embedded malware detection
2. To propose detection system towards recognising the presence of malware in the JPEG files.
3. To investigate the tolerance of accuracy in detecting the correct position of malware in JPEG files.

1.5 RESEARCH QUESTIONS

There are several research questions that can be addressed in this paper:

1. What is the current state-of-art in embedded malware detection?
2. Can we detect back the embedded malware in the JPEG image?
3. How effective the detection of embedded malware in JPEG image?

1.6 RESEARCH SCOPE

The form of steganography is wide which are including audio, video and image media. Thus, this project will narrow down the scope which is to implant malware in form of image media, i.e; JPEG. Then, we will focus on how to create an algorithm in order to create an infected JPEG image without corrupting the benign image file. We, then try to run and compare result to find the location of implanted malware in the infected JPEG files. The malicious content can destroy the system; therefore, by performing this analysis we can help Anti-virus developer to enhance their detecting system towards embedded malware.

1.7 PROJECT SCHEDULE

Project schedule can be referred in a Gantt chart in Appendix A

1.8 ORGANIZATION OF RESEARCH REPORT

Chapter 1 present the introduction of the research project. It start by presenting the project definition, project motivation, project objectives, project scope, project expected outcomes, project schedule and organization of the research project.

Chapter 2 provides literature review of the concept of steganography, the definition of computer malware as well as the concept of embedded malware.

Chapter 3 discusses the methodology used to perform the research project. It consists of the project tools to build and data flow diagram for general architecture of this project.

Chapter 4 discusses the findings and discussion of the project. It detailed the design of the research project and how analysis process is performed. Besides, this chapter will discuss about the way of locating the embedded malware in JPEG file.

Chapter 5 is a conclusion of this project. This chapter discussed about the problem encounter during the project and also given a recommended solution for the future studies.

1.9 CONCLUSION

In a nutshell, as the increasing of cybercrimes every day, we hope that this analysis will be able to facilitate the detection of embedded malware that are dangerous as they are capable of stealing the information of the system, as well as to damage the system function.

CHAPTER II
LITERATURE REVIEW

2.0 OVERVIEW

This chapter is about the precedence research that is done by the researchers. It starts with the concept of steganography followed by the definition of computer malware like Trojan, virus and worm. This chapter will end discusses with the definition of embedded malware and the present detection mechanism.

2.1 STEGANOGRAPHY

Steganography techniques is an art of hiding information in ways that obscure the detection of hidden message by conceal the existence of implanted information from anyone else except the sender and the intended recipients (Kumar, Pooja 2010). It transmits secrets through apparently innocuous covers in an effort to conceal the existence of a secret. In this cyber world, information like image, text, audio, or video, can be digitized, and advantage can be taken during the process to insert secret binary information into it.

Jókay *et al*. did mention that steganography use a media with fixed size to transfer. Therefore there are possibilities to estimate the maximum size of data to hide in the transfer medium. So, they describe the implementation of JPEG-based steganography where more information are inserted than the existing system, modifying the minimum number of discrete cosine transform (DCT) coefficient (Jókay, Moravčìk. 2010).

Chhajed *et al*. did review on binary image steganography and watermarking technique and they found that watermarking are more secure as spatial domain technique compared to other technique such as transform domain technique because it can retain the quality of the image (Chhajed, Deshmukh, Kulkarni 2011).

However, Meghanathan *et al*. also mentioned on the possibility of the hidden data analysed using steganalysis. Two types of image steganalysis: Specific and

Generic approach can be used to detect the presence of secret message and decode it. Specific technique depend on the existing algorithm used, high success rate means the ability to detect the message if the it hidden using the algorithm which the technique are meant for. Meanwhile, for Generic approach, it is independent to the existing steganography technique and can detect hidden message that use new and unconventional steganography algorithm (N. Meghanathan, L. Nayak 2010).

2.2 COMPUTER MALWARE

Malware is a piece of code that contain malicious intentions and performs processes that the user is not aware which changes the behaviour of either the operating system kernel or some security sensitive applications, without a user consent and in such a way that it is then impossible to detect those changes using a documented features of the operating system or the application. It is divided into several categories according to its goals and spread method (Slay, Turnbull 2006; Szor 2005) such as:

Viruses - A program that attempt to find other programs and infect them by attaching a copy of itself to them.

Worms - A self-replicating program that able to execute automatically and propagate itself across network without any help from the users and usually exploiting vulnerability.

Trojan Horses - Program that disguise itself as a legitimate tool with some useful functionality to attract users to download and run it on computer. This program contain hidden malicious process that running behind so that it cannot be traced and looks like a normal program.

Logic bomb – A malicious code that trigger at certain time pre set by attacker.

Exploit – A special code used to take advantage of vulnerability in software, usually have ability to escalate the privilege of a computer.

Downloader – A program used to download and install other malware from a remote source to a victim's host.

Virus generator – A tool to create viruses with specific features.

Spammer – Distributes fake information, usually via email, which contains email messages or web pages (**phishing**), and aims to collect users' private data or distribute misleading information.

Keylogger – record all users' activity involving keystrokes and mouse movements over a period

Rootkit – Used to hide the existence of a malware in a computer. It can prevent the operating system from showing evidence of a rootkit existence to the end-user.

Spyware – Usually, it is legitimate software but, which at the same time, collects an end-user's private information. Such software is usually used for marketing purposes.

Ransomware – unlawful forcing schemes as attackers hijack and encrypt the victim's computer and then demand a ransom from the victim for computer to be back in normal condition. (Xin Luo, Q. Liao 2007)

In related research, Siddiqui *et al.* did static analysis to extract the behavior from worms and clean programs in order to detect the presence of Internet Worm. They classify the dataset using the sequence of instruction extracted as the primary classification feature (Siddiqui, C. Wang, and J. Lee 2009). H. Binsalleeh *et al.* also did analysis on Zeus botnet by reverse engineer the crime toolkit to understand the architecture and the functionality of Zeus (H. Binsalleeh, T. O., Boukhtouta, Sinha, Youssef, Debbabi, Wang 2010). From there, they manage to come out with a tool to recover the encryption key used in communication between bot and Command and Control (C&C) server and also map the structure of the Zeus botnet network messages.

Several things that Zeus will do once it executed in the victims' computers are by copying itself to another location, execute the copy and delete the original file. Then it will lower Internet Explorer browser security settings by changing registry entries, injects code into other processes, main process exits, injected code hooks APIs in each process, steals several different type of credential found on the system, downloads config file and processes it, uses API hooks to steal data and lastly ends data back to C&C (Wyke 2011).

In current threat reported, Stuxnet, Flame and Duqu are considered as the state-of-the-art malwares as they are very sophisticated in architecture and remain undetected for such a long time. According to McDonald *et al.* Stuxnet 0.5 is believe as the earliest known Stuxnet version to be analyzed, discovered in

November 2007 and said to be in development as early as November 2005 (McDonald, L.O Murchu, S.Doherty, Chien 2013).

2.3 EMBEDDED MALWARE DETECTION

Typically, malware means a software program designed to damage or does any unwanted action on a computer system. Embedded means is enclosed in a surrounding mass. Thus, embedded malware can be describe as a software program that surrounded in a surrounding mass which intended to damage and does any malicious action on a computer system. The attacker implant the malicious code or file on the target host which cannot be detected by signature-based antivirus detectors even if a malware's exact signature is present in the detector's database (Stolfo, S. J., K. Wang and W. J. Li 2007).

Embedded malware is a new threat that brings a challenge towards security system. This is due to their ability to open their respective application software without its behaviors getting detected. Intelligent embedding can be further enhanced to allow automatic execution of embedded malware when the benign file is opened (Shafiq, M., Khayam, Farooq 2008). Shafiq et al come with a method to detect the embedded malware, by comparing the result Markov n-gram detector with the only known embedded malware detector using two different infected datasets. They manage to provide a significantly higher detection rate at the cost of higher false positive rates and argue that high accuracy can be achieved when the Markov n-gram detector is deployed in conjunction with commercial on the shelf antivirus software.

Abou-Assaleh *et al.* also justify earlier that n-grams could capture features that are specific for authors, coding styles, or even behavioral features. Since the captured features are implicit in the extracted n-grams, it would be difficult for virus writers to deliberately write viruses that fool n-gram analysis even when they have full access to the detection algorithm. They prove this by manage to achieves 100% accuracy on training data, and 98% accuracy in 3-fold cross-validation for 65 distinct Windows executable files consist of 25 malicious code and 40 benign code that extracted from e-mail messages and ranging of size from 12.5 to 420 KB (Abou-Assaleh, N. C., Keˇselj, Sweidan 2004).

2.3.1 TECHNOLOGY IN EMBEDDING

There are several tools that can be used to embed file into another file.

2.3.1.1 *Steghide* (2013)

Steghide is a free software for steganography. It allows users to hide embed data in images (bmp, jpeg) and audio files (wav, au) by changing some bits say "least significant", encrypted or not encrypted. The files containing the secret data are not altered in appearance as other people will see the same image and hear the same sound. *Steghide* platforms are available for Linux and Windows and works by command line.

Features include the compression of the embedded data, encryption of the embedded data and automatic integrity checking using a checksum. The JPEG, BMP, WAV and AU file formats are supported for use as cover file. There are no restrictions on the format of the secret data.

2.3.1.2 *OpenPuff* (2013)

OpenPuff is a portable steganography and marking software for Windows, enables users to hide sensitive data inside images, audio files and videos. It support various formats of files such as BMP, JPG, PNG, MP3, WAV, MP4, MPG, FLV, SWF, PDF and more. *OpenPuff* supports 512bit key cryptography with SHA512 password extension as well as data scrambling, data whitening and creation of decoy data.

2.4 CONCLUSION

Based on the literature review of this research, we have gain the brief idea about the concept of steganography, types of embedded malware, and the technologies that is used to embed files and in order to discover the presence of embedded and hidden threat.

CHAPTER III
RESEARCH METHODOLGY

3.0 OVERVIEW

This chapter explains the data preparation for this research as well as the project tools.

3.1 DATASET PREPARATION

For this research, we will use a collections dataset that consist of 100 JPEG files and 100 malware executables. For detection mechanism, it will be based on statistical profile on n-gram.

3.1.1 JPEG FILES

JPEG is a compression algorithm optimised for photographic images, is widely use in imaging device such as digital camera, standalone or embedded in smart phones, something we encounter on a regular basis. JPEG is not limited to a certain amount of color as it can compress 24 bits per pixel that is equal to 16 million colors and is popular due to its variable compression range, meaning that you're able to more easily control the amount of compression, and consequently, the resultant image quality. We are using JPEG files for this research as by its degree of "lossiness" that that user can control by adjusting compression parameters. Therefore, we can achieve any size of files with needed amount of quality.

Resolution	8 megapixel, 4:3
ISO	400
Exposure value	0
Contrast	Normal
Saturation	Normal
Size	1.3 – 1.7 MB

Table 3.1: Camera setting to generate JPEG files

3.1.2 MALWARE

Most of our malware samples are obtained from (VirusSign 2013) and we manage to get about 500 malwares. The samples comprise of

Trojan Horse	235
Worm	79
Virus	186

Table 3.2: Malware category of the samples

However, when we run our sample to *Microsoft Security Essential*, antivirus software from *Microsoft Corporation*, we detect some differences in the definition compared to reports given by VirusSign. Some of the sample are detected as worm such as *PWS:Win32/Zbot*, *Worm:Win32/Clisbot.A*, and *Worm:Win32/Rebhip.A*.

For this experiment, we will pick 100 samples from the malware samples as our aim is to embed one jpeg file with one malware sample.

3.1.3 N-GRAM

N-gram is data mining technique used to define a sequences of n items a given data. An n-gram of a sequence is a normalised frequency histogram (or the distribution) of n bit symbols in the sequence. In this methodology, we will introduce this technique in our experiment.

3.1.4 FRAMEWORK

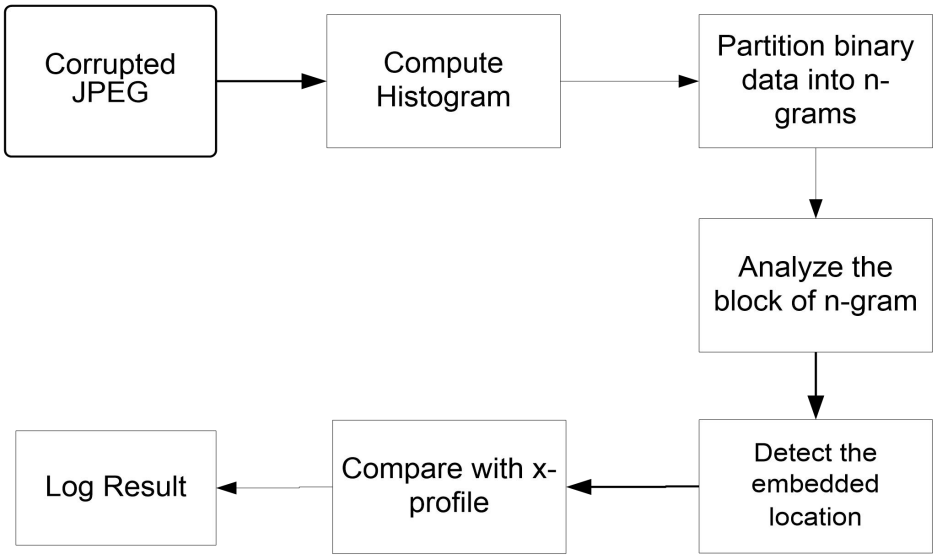

Figure 3.1: Flowchart for detection mechanism of embedded malware in JPEG file

The detection mechanism will be implemented in the framework to find the embedded malware in JPEG file.

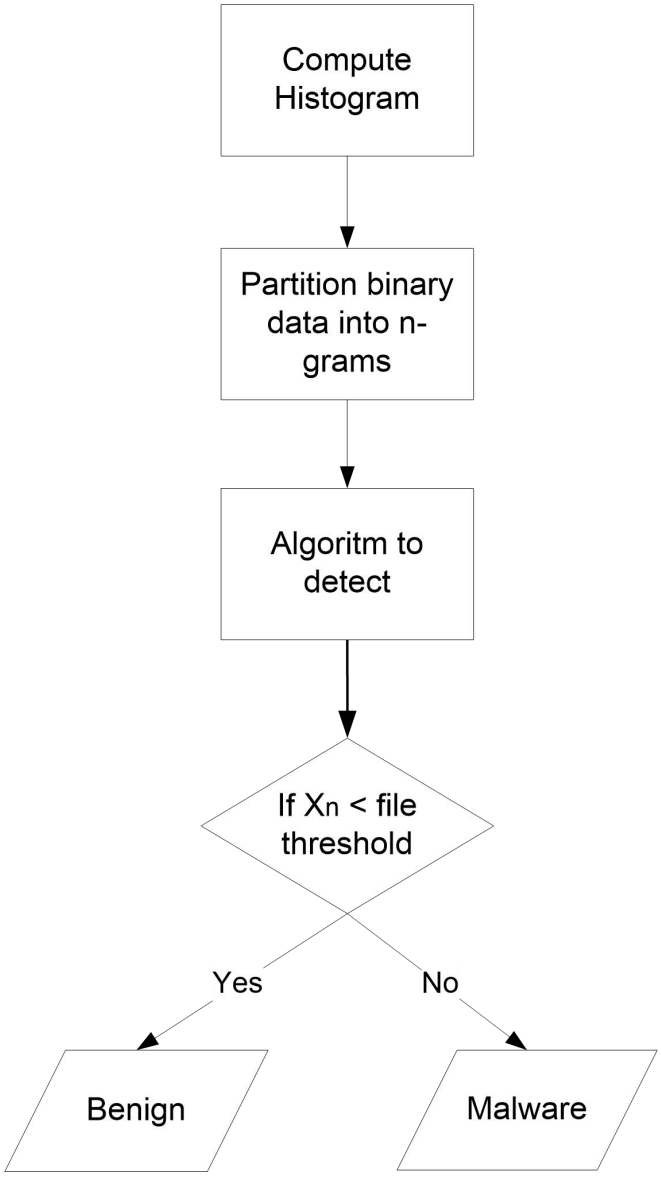

Figure 3.2: Analysis and detection for embedded malware based on statistical profile on n-grams.

Statistical data is retrieved from this binary partition, containing the n-grams' degrees of closeness to both the malware and benign profiles. Training data is a collection of data that is passed to detection system, so it can begin learning it to find "response value". The decision component relies greatly on the information learned from training data, contained in these profiles, and uses a threshold to make the decision. If the executable's profile is below this threshold, it is characterises as benign or else it is a malware.

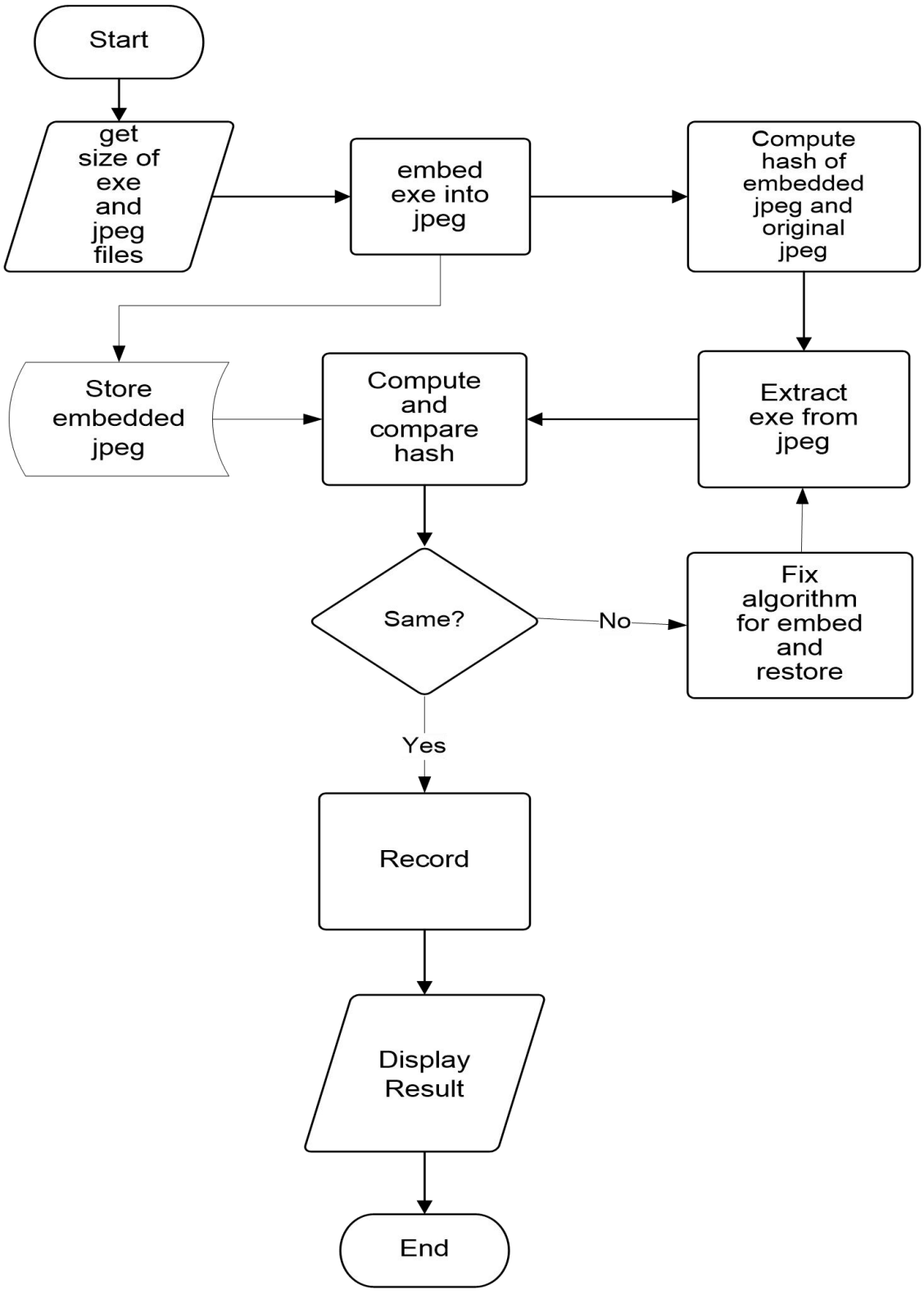

Figure 3.3: The structural work of the experiment

3.2 PROJECT TOOLS

Tools that have been used in order to perform this research project are as followed:

Software/Tools	Description
Visual C# Express 2010	Programming language used to develop the algorithm and the program.
Notepad	To record the infected JPEG details.
Microsoft Office Visio 2007	Tool to design the data flow diagram of the research project.
Microsoft Office Excel 2007	Tool to create graphs and read logs for analysis.
Nikon Model: D90	Tool to prepare the benign JPEG image.

Table 3.3 Project tools used for research

3.3 CONCLUSION

In this chapter, the methodology used to perform the research project is discussed. Also, tools that have been used are listed on the table 3.3. Our hope is to make sure the dataset is ready for experiment as to prove that the possibility to detect the embedded malware, hidden in the JPEG files.

CHAPTER IV

FINDINGS AND DISCUSSIONS

4.0 OVERVIEW

This chapter discussed on the detection system towards recognising the presence of malware in the JPEG files and to investigate the tolerance of accuracy in detecting the correct position of malware in JPEG files

4.1 INFECTED DATASET

The algorithm is completed by developer itself. The testing is accomplished by combining the malware binary into JPEG binary. Fundamentally, there are certain criteria that must be followed in order to determine the algorithm is successful created. Those criteria are:

- JPEG image must not corrupted
- The insertion of the malware must be at random location (but, within the content of the image).
- The hash value of the original image and the infected JPEG image must not same.

If the infected JPEG image fulfills all the criteria, we can conclude that the insertion is successful. The malware is successful implanted in the benign file.

4.1.1 RESULT

FIGURE 4.1.0 Original image

FIGURE 4.1.1 Infected image

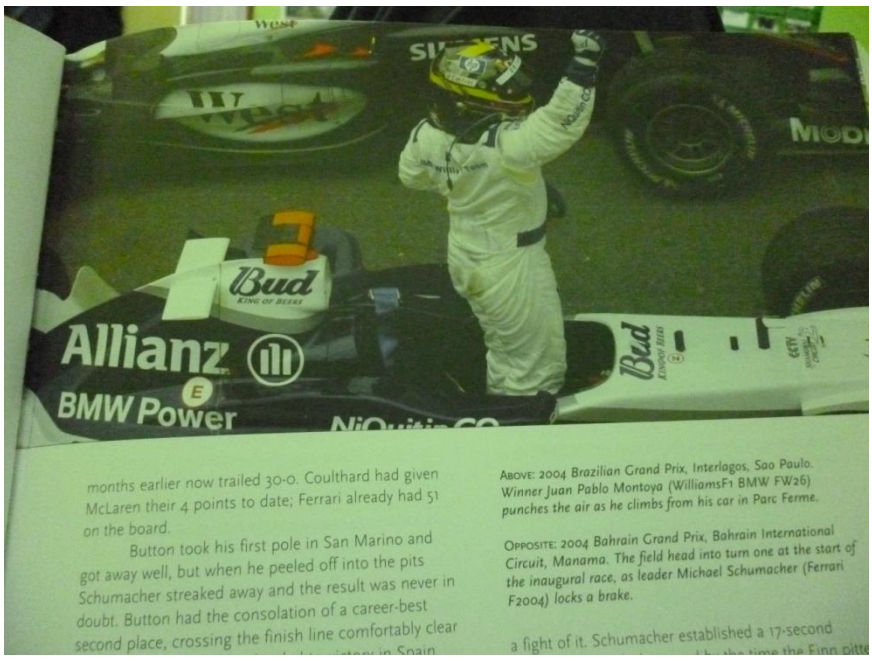

FIGURE 4.1.2 Original image

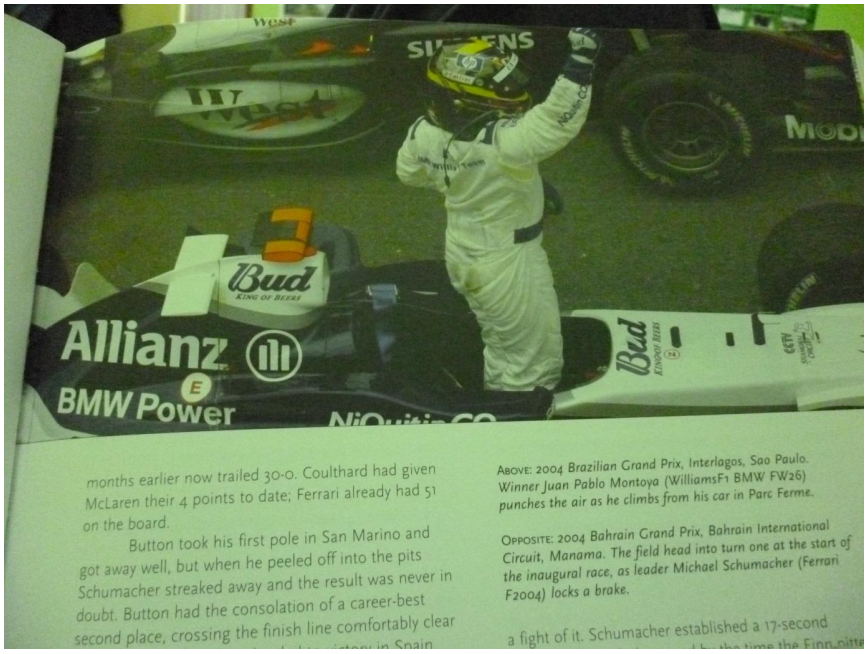

FIGURE 4.1.3 Infected image

As we can, the benign file is not corrupted. Therefore the location of the embedded malware has not disrupts the binary of the benign file. Based on the experiment that has been conducted, the best range location of insertion malware binary is at range 300 before the end of JPEG byte count.

4.1.2 EMBEDDED ANALYZER REPORT

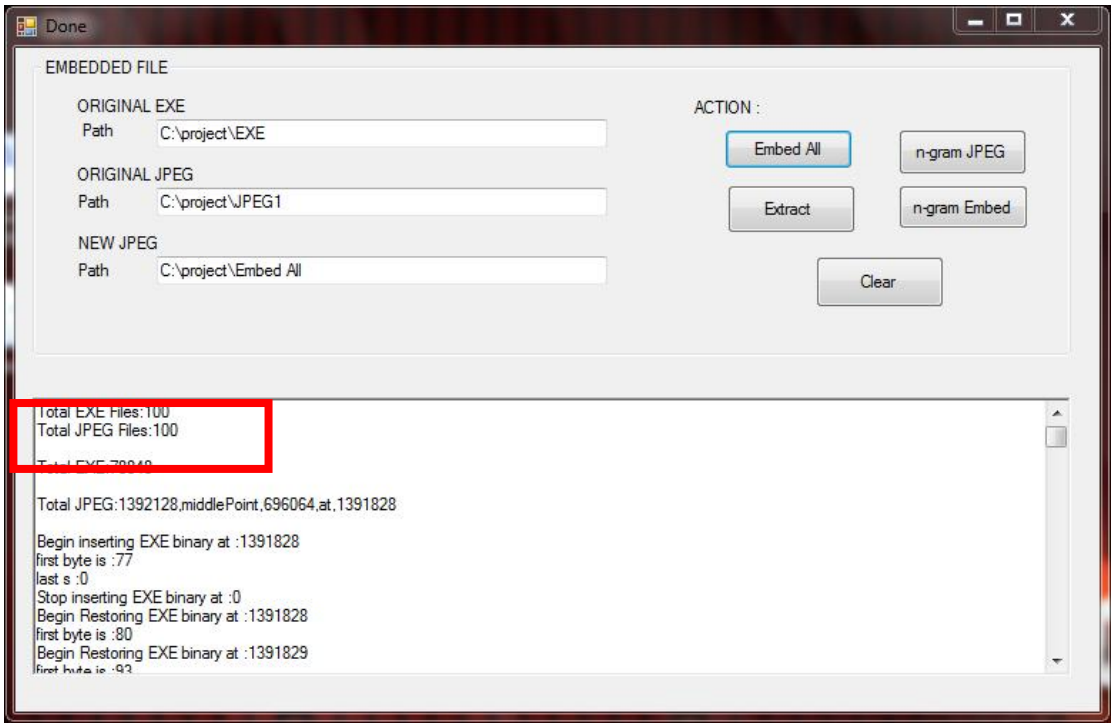

FIGURE 4.1.4 Infected image report

100 EXE files and 100 JPEG files has been executed. We can check the details of the process in the rich textbox output. From our collection, we embed first malware into first JPEG file, second malware into second JPEG file and this sequence continue until it reach the last file for both malware and JPEG file.

shortFileNames	THRESHOLD	DIVIDEOPERATOR	AllEmbed.Length	Counter	Estart[fileCounter - 1]	Estop[fileCounter - 1]	std
1.JPG	8	2	1470976	1470975	1391828	1470676	147.2001698
10.JPG	8	2	1602102	1602101	1555668	1601802	107.6093862
100.JPG	8	2	1520989	1520988	1486548	1520689	124.5219659
11.JPG	8	2	1363783	1363782	1341652	1363483	139.5733499
12.JPG	8	2	1484998	1484997	1397972	1484698	150.4449733
13.JPG	8	2	1461248	1461247	1443028	1460948	167.4938506
14.JPG	8	2	1611324	1611323	1589460	1611024	134.7863124
15.JPG	8	2	1510400	1510399	1493716	1510100	78.50840719
16.JPG	8	2	1538048	1538047	1503956	1537748	196.0539977
17.JPG	8	2	1419776	1419775	1359060	1419476	48.96856134
18.JPG	8	2	1411208	1411207	1376980	1410908	201.0997762
19.JPG	8	2	1395712	1395711	1342164	1395412	151.3641635
2.JPG	8	2	1535137	1535136	1469652	1534837	98.76735291
20.JPG	8	2	1454616	1454615	1438420	1454316	197.9593898
21.JPG	8	2	1384269	1384268	1322196	1383969	230.9885928
22.JPG	8	2	1331712	1331711	1320148	1331412	90.38368216
23.JPG	8	2	1477184	1477183	1434324	1476884	148.7147605
24.JPG	8	2	1400320	1400319	1323220	1400020	133.3142153
25.JPG	8	2	1437696	1437695	1393876	1437396	27.22755957
26.JPG	8	2	1454592	1454591	1409236	1454292	201.9375894
27.JPG	8	2	1454568	1454567	1438420	1454268	139.6480576
28.JPG	8	2	1506304	1506303	1412308	1506004	185.2286695
29.JPG	8	2	1484288	1484287	1415892	1483988	157.4201067

FIGURE 4.1.5 Snippet of Summary Report

4.13 LOGFILE OF THE EMBEDDED PROCESS

	A	B	C	D	E	F	G	H	I
1	Total EXE	Total JPEG	Start Embed at	Stop Embed at	Original Hash	Embedded Hash			
2									
3	78848	1392128	1391828	78848	99FD09B3D2247E4BA7B51D27	A5A5B162A3F5B648F5D32119F34E33			
4	46134	1555968	1555668	46134	CE43BA188157CA4A8415621B	3F3626F584BF9C7C5D13B845FC47D8			
5	34141	1486848	1486548	34141	C5A2C29EE820116D8F3FA280	C9D0CA002F897083B1103419F3EB4C			
6	21831	1341952	1341652	21831	944047A6DFE25BB4DA9304A3	D7F75FC12A30B3F3E05DE4A58FC563			
7	86726	1398272	1397972	86726	6209213FE8FAB103B48400E7C	C0B7473EECBC364773BA1A971F5931			
8	17920	1443328	1443028	17920	B61722AF7D4D90CBC15A0854	510A0329D512390FBE0083D5E15493			
9	21564	1589760	1589460	21564	57FF5E3D3F673053F9BCBA7A	43CBDE986AAE8EB9957015C4EAB4E7			
10	16384	1494016	1493716	16384	5C26A53CE5B3DED0B682D155	ABEA33D3B88A8107CC7D94F31A113			
11	33792	1504256	1503956	33792	29CBDB6CA61931BECCFFB68E	97EFF588DE6621C0C63A4C39BA517F			
12	60416	1359360	1359060	60416	14910FCA4F4171741CF33386C	D72308EBCA96C7B49AD14CA84C640			
13	33928	1377280	1376980	33928	537D824D9EBC8C6CF2444FCB	ACA88850B052B9CB75571566E673A5			
14	53248	1342464	1342164	53248	B7E99D9218A00460D8DB6A1C	1D1B2E6008013717DFB550CD56C695			
15	65185	1469952	1469652	65185	84426544EA63A8820E08E0C3C	574C5E58D9940976A0CDFDE7452E10			
16	15896	1438720	1438420	15896	0B399AD14AFFFC4DF9E5EC46	9669B83CE590A1EEED07380C9ED62A			
17	61773	1322496	1322196	61773	4EFEB15101A0F7C60E55E23CC	34262475BF764C5AD3B46FEE6B5680			
18	11264	1320448	1320148	11264	8ADE34DCA9FB1E3DF7CD2AC	780E717BACF62188A8D93868F2781A			
19	42560	1434624	1434324	42560	77AB11C1AFEC449206052773C	ABAB9484EEA38448E46EB941158FAA			
20	76800	1323520	1323220	76800	EC18C41E367F7B418F0AFA7D	841430859559D441C33508F5B88C45			
21	43520	1394176	1393876	43520	E30A60CB9C1CA99147022A55	2122F699A746C6A652F5FE2CAF8436			

FIGURE 4.1.5 Log file

All the process has been recorded in the embedData.cvs and the arrangement in the Microsoft Excel. The element that being stored are the length of malware, length of JPEG file, the start location of insertion malware, the last location that being insert in the JPEG file, original JPEG's hash value, and embedded JPEG's hash value.

The purpose of storing that information is to evaluate either the embedded process insert the malware at random location or not. Due to that, we record the starting point for each insertion process. Also, we verify the hash value of the original image and the infected image in order to ensure the embedding process has occurred.

4.2 N-GRAM

An n-gram is a sub-sequence of n items from a given sequence. Usually, *n*-grams are non-overlapping sequences of items but can be designed to be overlapping. We choose to generate the n-gram by dividing the bytes per n-value. For this experiment, we run with value n equal to 100, 500 and 1000. The *n*-grams of the malware and benign programs are obtained and gathered as two separate collections. We will use the n-gram for determining the location of embedded malware in JPEG file.

For this analysis, we take the result of the first JPEG file and first malware in our collection.

4.2.1 n = 1000

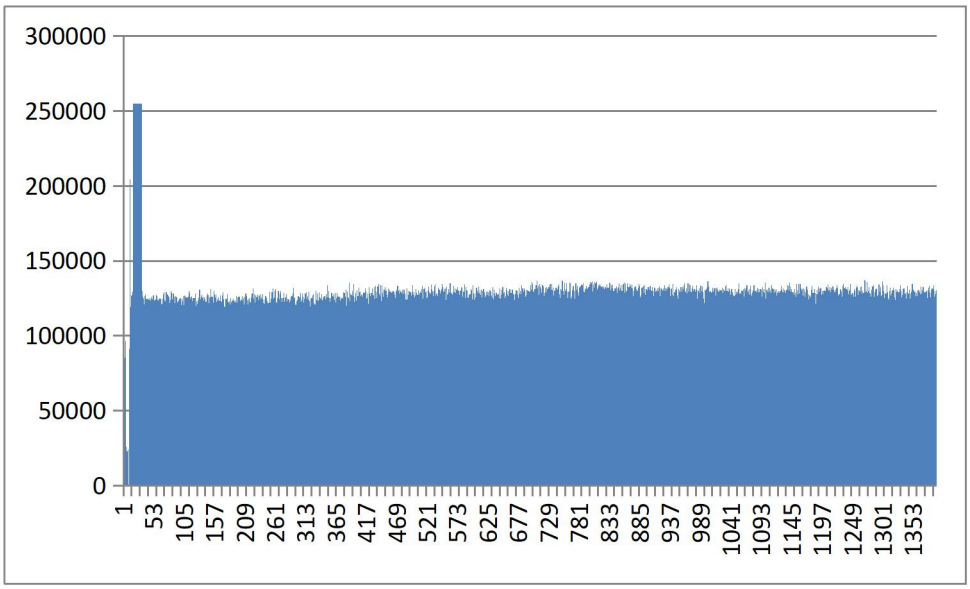

FIGURE 4.2.0 N-gram JPEG Original

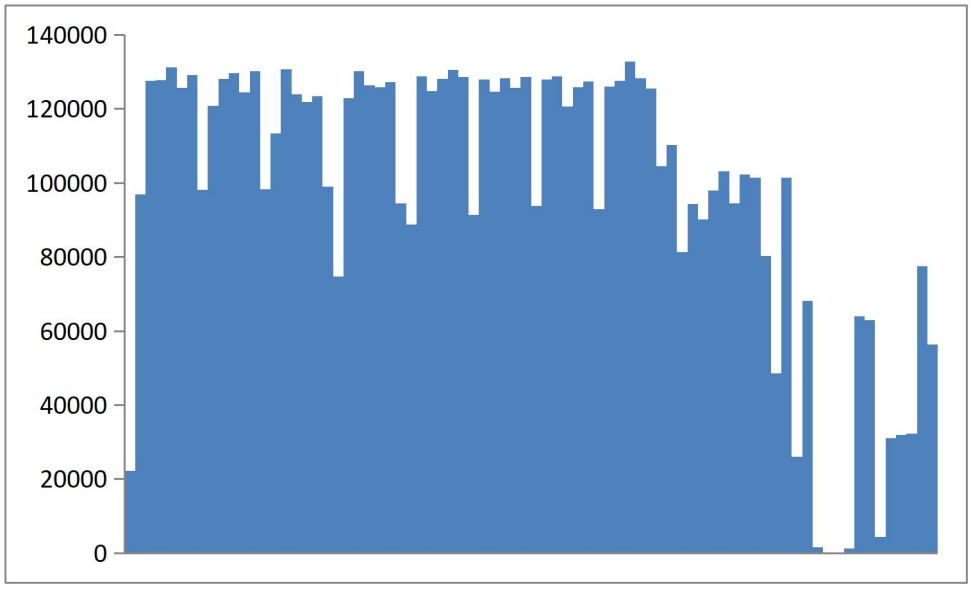

FIGURE 4.2.1 N-gram Malware Original

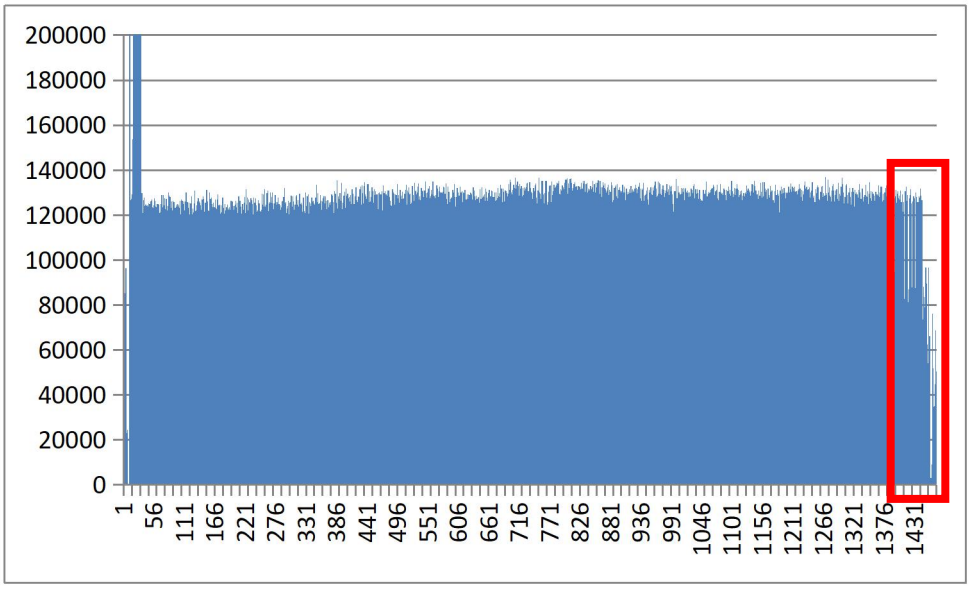

FIGURE 4.2.2 N-gram Embedded Malware

4.2.2 n = 500

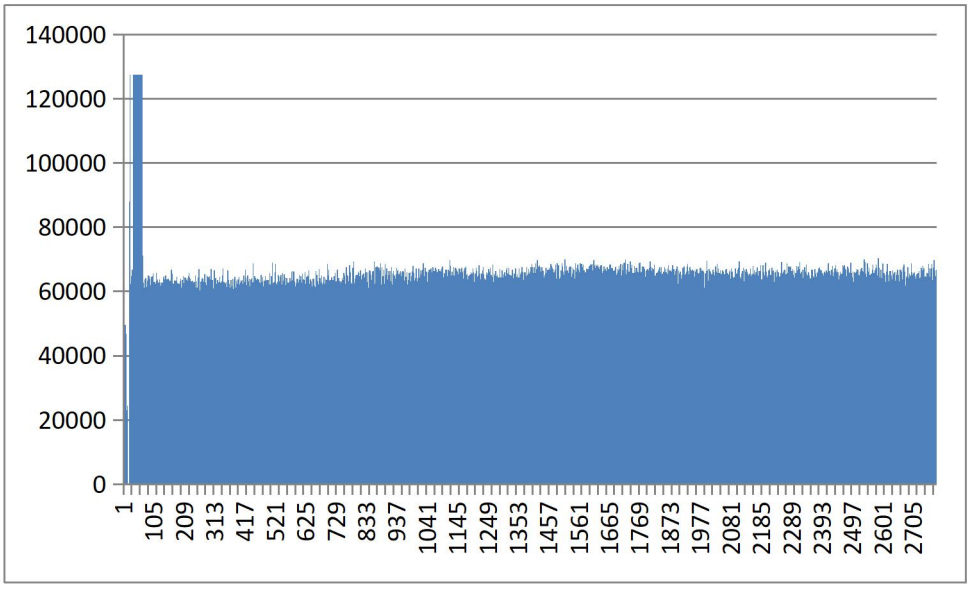

FIGURE 4.2.3 N-gram JPEG Original

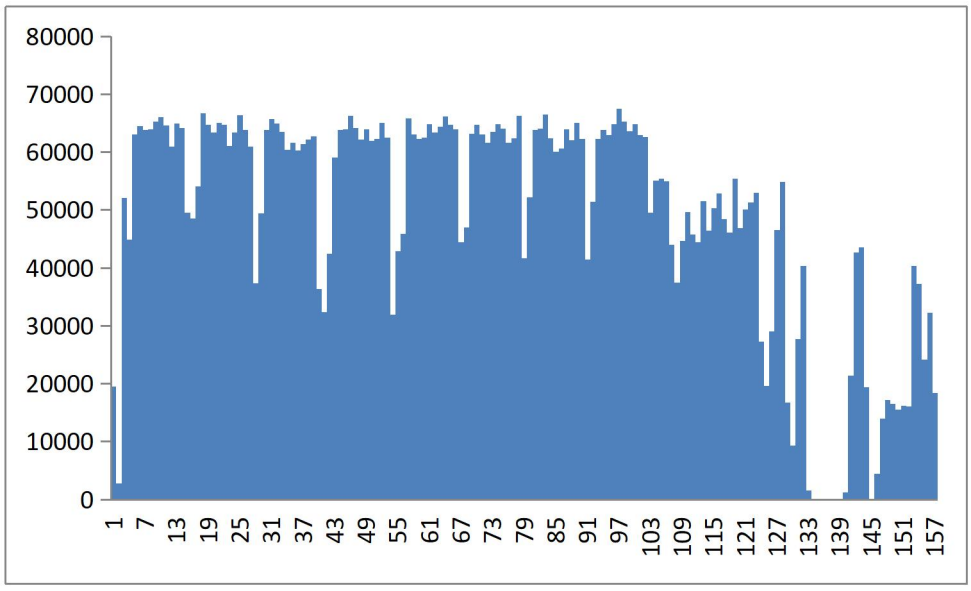

FIGURE 4.2.4 N-gram Malware Original

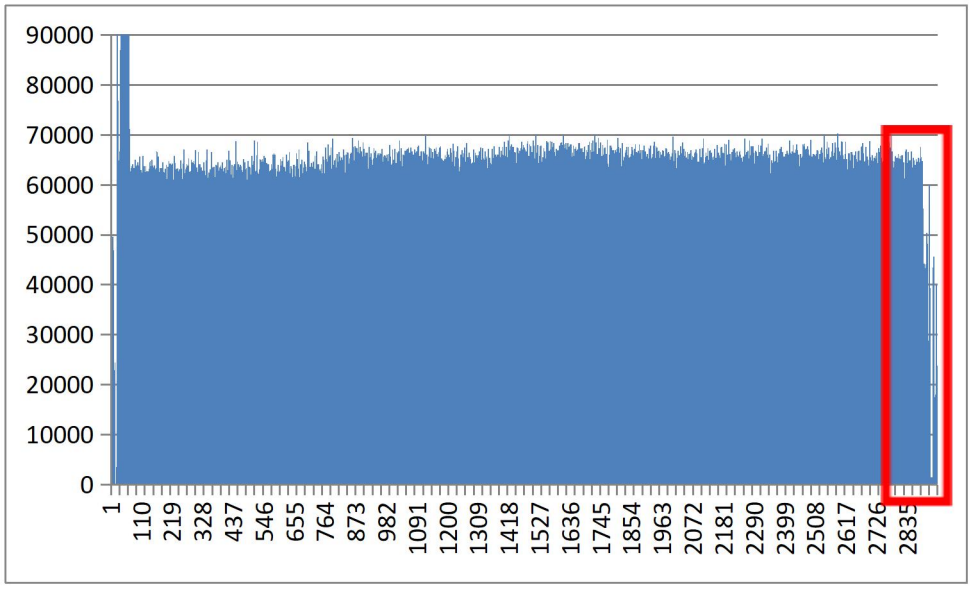

FIGURE 4.2.5 N-gram Embedded Malware

4.2.3 n = 100

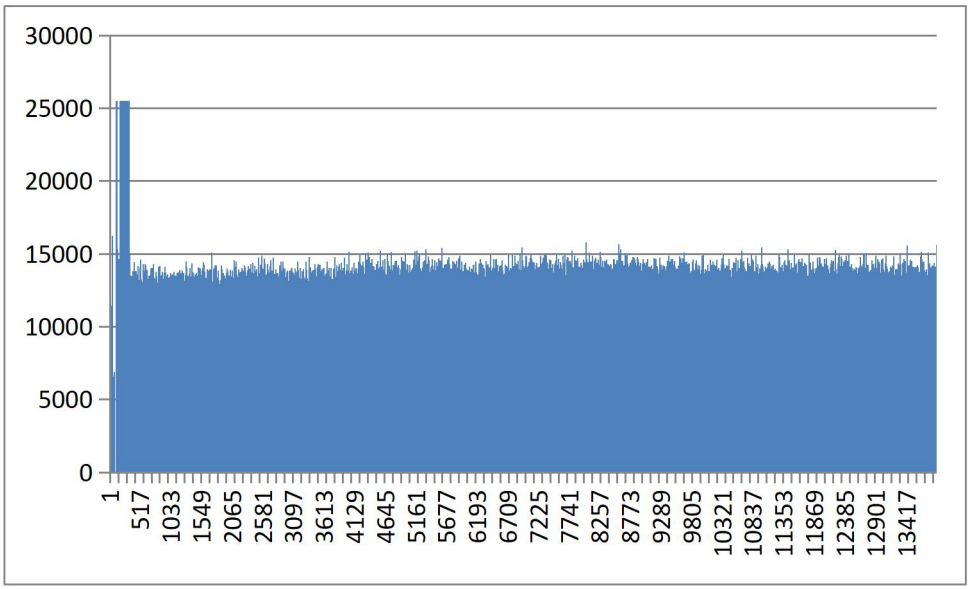

FIGURE 4.2.6 N-gram JPEG Original

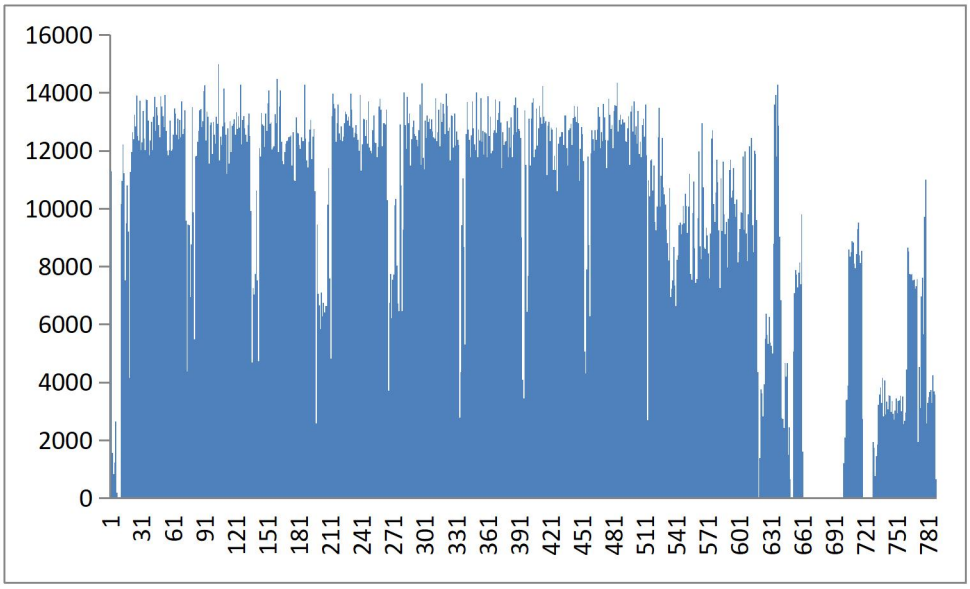

FIGURE 4.2.7 N-gram Malware Original

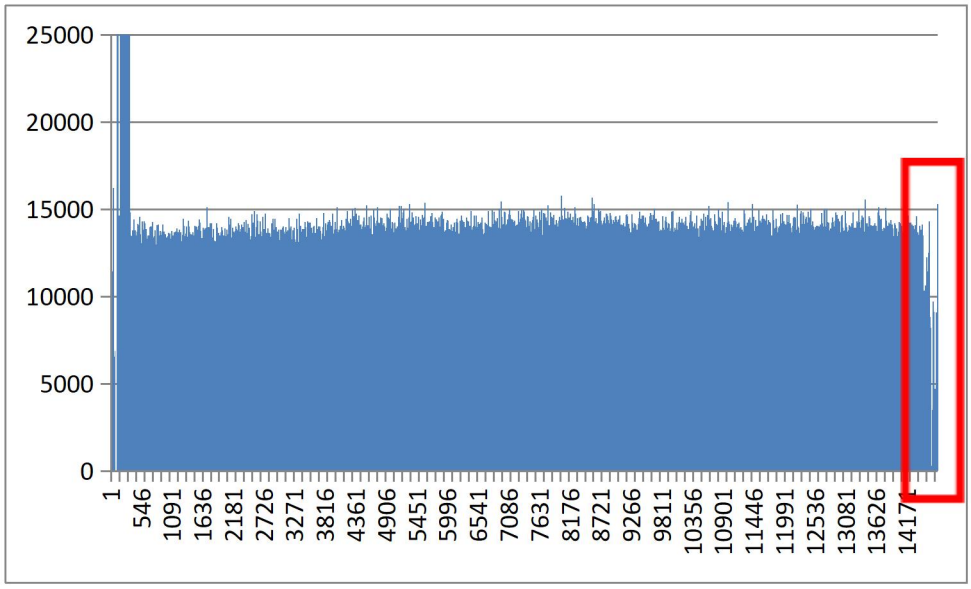

FIGURE 4.2.5 N-gram Embedded Malware

By rough observation, we can see that at the end of the n-gram for all, there are big perturbation of bytes occur. As the value of n is decrease, we can see the perturbation of bytes at the end of the n-gram as compare to the n-gram of original JPEG file. Still, we need to locate the exact position of the malware rather than roughly assume the position.

Then, for each file that is embedded with malware, after finish generates its n-gram, for each window we generate several values to determine the location of malware. The values are the minimum and maximum value of bytes, its mean, median, variance and standard deviation.

From there we do comparison that is for each window, we compare the value of standard deviation, *std* with Perturbation Value = Average (Standard Deviation of previous file + Threshold). If value *std* is less than Perturbation Value, we are confident to say that is the location of embedded malware. For each n-gram, we manage to estimate whether malware exist on each windows.

From the graph below, we can see the estimated location of malware in the embedded JPEG file:

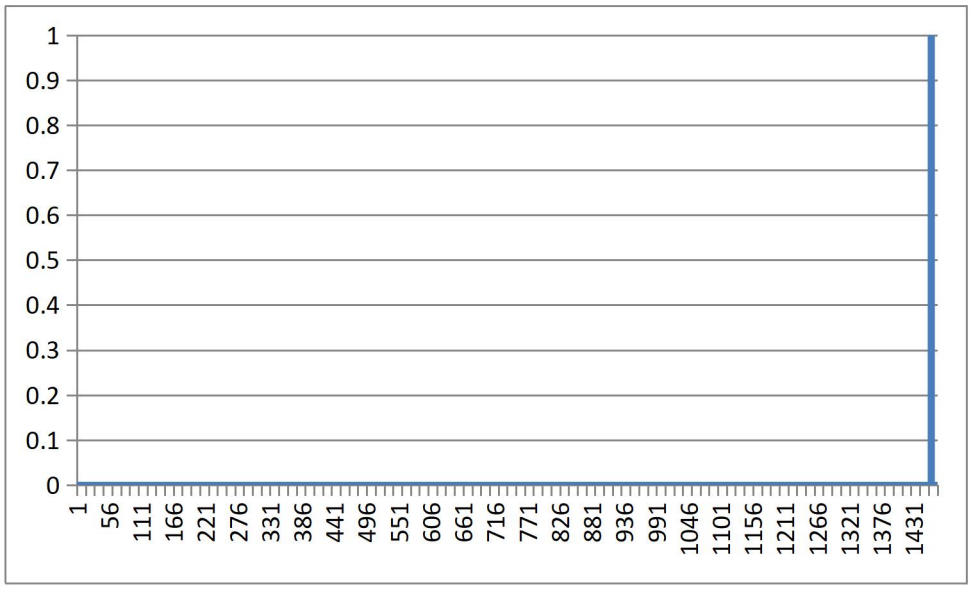

FIGURE 4.2.6 Malware location for n-gram = 1000

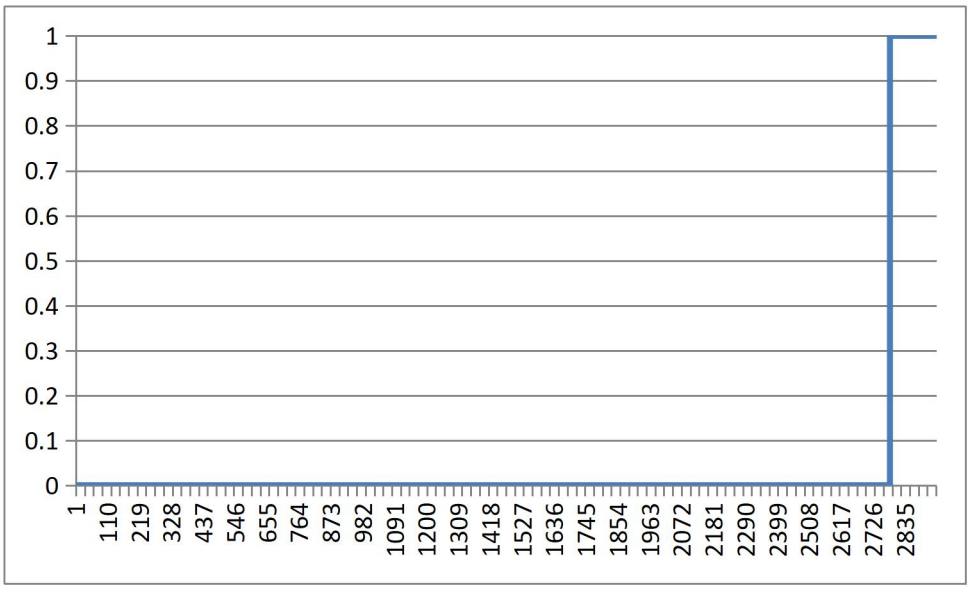

FIGURE 4.2.7 Malware location for n-gram = 500

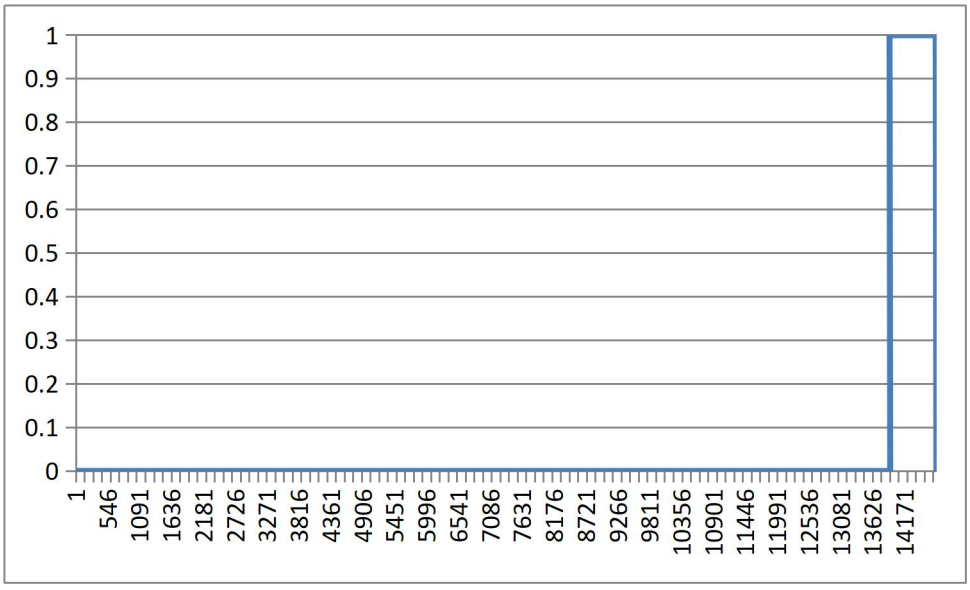

FIGURE 4.2.7 Malware location for n-gram = 100

From the analysis, we can figure that the location of the malware is at the end of the file based on the projection in the n-gram and the perturbation value generated from experiment.

CHAPTER V

CONCLUSION

5.0 ISSUES AND CHALLENGES

The most challenging part while executing this research is to collect the malware dataset. It is hard to find websites that provides malware code for research especially after the most popular site VX Heaven.com has been taken down by Ukraine government. Luckily there are some malware sample collection like VirusSign.com that used by tester for antivirus industry. There are a lot of malware collections, though; the collection of the new malware is modest. So, we collected the malware that has been release in a few years back as our dataset. Besides, the issues of embedded malware still not widely distributed, hence, the algorithms or tools to generate an embedded malware are hard to find.

5.1 RECOMMENDED FUTURE WORK

We would highly recommend any person willing to improve on this type of project research by extracting the embedded malware from the JPEG binary. Thus, a secure JPEG image can be produced and can be used safely without being tempting by any malicious code.

REFERENCE

Kumar, A., K. P. (2010) 'Steganography- A Data Hiding Technique'. *International Journal of Computer Applications (0975 – 8887)*, Vol.9 (No.7).

McDonald, G., L.O Murchu, S.Doherty, Chien, E. (2013). "Stuxnet 0.5: The Missing Link", Symantec Corporation.

K. V. D. Gyankamal J. Chhajed, Trupti S. Kulkarni (2011) 'Review on Binary Image Steganography and Watermarking'. *International Journal on Computer Science and Engineering (IJCSE)*, Vol.3.

H. Binsalleeh, T. O., A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, L. Wang (2010) 'On The Analysis Of The Zeus Botnet Crimeware Toolkit'. *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference*. Ottawa, ON, pp 31- 38.

Wyke, J. (2011). 'What is Zeus? ', SophosLab, Oxford, UK.

Slay, J., Turnbull, B. (2006). "Computer security education and research in Australia". IEEE Security and Privacy 04, No. 05, 60-63.

Shafiq, M., S. Khayam and M. Farooq, (2008) 'Embedded Malware Detection using Markov n-grams', Zamboni, D. (ed. *DIMVA '08 Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer-Verlag Berlin, Heidelberg ©2008, pp. 88 - 107.

Jókay, M., Tomáš Moravčìk. (2010) 'Image-Based JPEG Steganography'. *Tatra Mountains Mathematical Publications*, Volume 45 (Issue 1). pp 65–74.

Siddiqui, M., M. C. W., J. Lee (2009) 'Detecting Internet Worms Using Data Mining Techniques'. *Journal on Systemics, Cybernetics and Informatics*, Vol. 6 (No.6). pp 48-53.

Meghanathan, N., L. Nayak (2010) 'Steganalysis Algorithms For Detecting The Hidden Information In Image, Audio and Video Cover Media'. *International Journal of Network Security & Its Application (IJNSA)*, Vol.2 (No.1).

OpenPuff (2013) Retrieved 29 April 2013, from http://embeddedsw.net/OpenPuff_Steganography_Home.html.

Szor, P. (2005). "The Art of Computer Virus Research and Defense". Addison Wesley Professional.

Bateman, P. (2008). "Image Steganography and Steganalysis". Department of Computing, Faculty of Engineering and Physical Sciences, University of Surrey, UK.

Steghide (2013). Retrieved 29 April 2013, from http://steghide.sourceforge.net/.

Stolfo, S. J., K. Wang and W. J. Li (2007). "Towards Stealthy Malware Detection". Department of Computer Science, Columbia University.

Abou-Assaleh, T., N. C., Vlado Kešelj, Ray Sweidan (2004) 'N-gram-based Detection of New Malicious Code', *COMPSAC '04 Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts*. IEEE Computer Society Washington, DC, USA ©2004, pp. 41-42.

VirusSign (2013). Retrieved 15 April 2013, from http://samples.virussign.com/samples/.

Xin Luo, Q. L. (2007) 'Awareness Education as the Key to Ransomware Prevention'. *Journal Information Systems Security*, Volume 16 (Issue 4). pp 195-202.