Scientific Research and Essays Vol. 5(15), pp. 1965-1977, 4 August, 2010
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 ©2010 Academic Journals

Full Length Research Paper

Novel approach for high (secure and rate) data hidden within triplex space for executable file

A. A. Zaidan*[1], B. B. Zaidan[1], O. Hamdan Alanazi[2], Abdullah Gani[2], Omar Zakaria[2] and Gazi Mahabubul Alam[3]

[1]Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Selangor Darul Ehsan, Malaysia.
[2]Faculty of Computer Science and Information Technology, University of Malaysia, 50603, Kuala Lumpur, Malaysia.
[3]Faculty of Education, University of Malaya, Malaysia.

Accepted 2 July, 2010

Steganography is an art that involves the concealing of information with the aim of making the communication invisible. Unlike cryptography, where the main aim is to secure communications from the Snooper by encoding that data in such a way that it becomes visible but not understood. In this paper, we propose a framework that will integrate both steganographic and cryptographic approaches within a hyiperd space of an EXE file. This approach will be used for highly securing data that is hidden; using both the advance encryption standard (AES) method and statistical technique (ST) computation. To further address the security issues, we will use the EXE as a cover for the hidden data; the executable file cover was selected in this approach because of its ability to hide huge amounts of data within a hyiperd space of an EXE file which in turn overcomes the problem of the data hiding quantity. The use of the EXE file also makes sure that the changes made to it will not be detected by anti-virus and its functionality will not be negatively affected after the hiding process, features of the short-term responses were simulated, and indicated that the size of the hidden data does depend on the size of the Unused Area1+ Unused Area 2 + Image Pages; within the cover file which is equal to 28% of the size of the exe.file before the hiding process. Most antivirus systems do not allow direct writing into executable files, so the approach of the proposed system is to prevent the hidden information from being disclosed by these systems and the exe.file still functioning normally after the data hiding process.

Key words: Steganography, hidden data, encryption, advance encryption standard (AES).

INTRODUCTION

Steganography is the art of concealing data during transmition through seemingly innocent carriers in an effort to obscure the existence of the data. The literal meaning of the word Steganography is covered or hidden writing as derived from Greek. Steganography plays a substantial role in security (Ahmed et al., 2010). However, it does not intend to replace the role of cryptography but supplement it. In concealing a message using Steganographic techniques lessens, the chance of a message being detected if the message is also encrypted, offers another layer of protection (Anckaert et al., 2005).

Therefore, some Steganographic methods integrate both traditional Cryptography and Steganography; where the initiator encrypts the secret message before the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover (Artz, 2001). Several terminologies have been specifically developed for use in the field of Steganography (Bender and Morimoto, 1996). The adjectives 'cover', 'embedded' and 'stego' were defined at the information hiding workshop held in Cambridge, England. The term "cover" refers to the description of the original and innocent massage, data, audio, video, and so on. Steganography is not a new science as it is quite ancient (Bassia et al., 2001), it has been used from time in memorial by ordinary people, spies, rulers, governments, and armies

*Corresponding author. E-mail: aws.alaa@gmail.com.

(Cvejic, 2002). There are many accounts about Steganography (Cvejic, 2004). For instance in ancient Greece, the belly of a share (a kind of rabbit) or pigeon was used for hiding messages using invisible ink (Yeh et al., 1999). In addition to this, ingenious methods such as shaving the head of a messenger and then tattooing a message or picture on the messenger's scalp were used (Cvejic, 2005). After the hair grew back, the message would be concealed until the head is shaved again, while the Egyptian employed a different approach which involved the use of illustrations to conceal messages (Lee et al., 2000).The information that is hidden in the cover data is referred to as the "embedded" data, and information hiding is a general term which is made up of several sub disciplines (Noto, 2001). The term deals with a wide range of problems beyond that of just embedding the contents in a message. The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret. Information hiding is a technique of hiding secrets using redundant cover data such as images, audios, movies, text documents, and so forth. Today this method has taken a center stage in a number of application areas (El-Khalil and Keromytis, 2004). Instances of digital video, audio and images are progressively being embedded with imperceptible marks, which may contain hidden signatures or watermarks that assists in the prevention of illegal duplication (El-Khalil and Keromytis, 2004). Thus Steganography is the process of inserting secret messages into a cover file, so that the existence of the messages is not apparent (Lee et al., 2000).

Research in information hiding has tremendously increased during the past decade with commercial interests as the driving force (Ahmed et al., 2010).

Typically, the amount of hidden information that is embedded in media such as image and music files is limited. Thus these types of embedding methods can easily be under surveillance from system managers in an organization that needs a high level of security. This has led to a research on new hiding techniques and cover objects in which hidden information can effectively be embedded. As a result of this research, the technique of embedding information in executable files has been explored.

Stilo and Hydan are deemed to represent common techniques for the embedding of information in executable files (Anckaert et al., 2005; El-Khalil and Keromytis, 2004). These techniques modify original files, thus allow code signing techniques that guarantee the integrity of the code to be used for the detection of hidden information (Anckaert et al., 2005; El-Khalil and Keromytis, 2004). But since the level of using computers and the computing environment has spiraled upwards, it makes the use of executable files to become a common phenomenon. However, the use of code signing techniques is not yet common (Anckaert et al., 2005; El-Khalil and Keromytis, 2004). Silo and Hydan modify program binaries that have been in optimization, so as to reduce the performance levels of the program binaries (Anckaert et al., 2005). Besides this, the amount of information to be embedded in executable files when Silo and Hydan methods are applied is limited; below 15% of the total cover size because these tools determine the number of bytes to be hidden based on the size of the program binaries (El-Khalil and Keromytis, 2004).

Aos and Bilal modify programs by implementing a system that commutates embeded information using both cryptography and steganography within an unused area 1 of an exe.file (Zaidan et al., 2009).

Ahmed and Aos devised a new technique that modifies hidden data in the Unused Area 2 within an exe.file using computation that involve Cryptography and Steganography (Naji et al., 2009). The aim of these two studies was to find a secure solution for a cover file without changing its size (Naji et al., 2009).

However, this research did not look into the anti-virus detection of the functionality of the exe.file; they were not able to tell if the exe.file is still functioning or not and whether the size of the information hiding is still limited (Ross, 1998).These are considered to be the main challenges for hidden data in the executable files. Hence, there is a need to carry out research on new hiding techniques that take into consideration the efficiency of a program using computation which involves cryptography and steganography.

In order to deal with the above mentioned challenges of the executable file when it is used as a cover, in this study, we examine new methods that consider the efficiency and the amount of information to be hidden. We also make sure that changes made to the exe.file will not be detected by anti-virus and the functionality of the exe.file is not affected negatively. In addition to this, we discuss the analysis techniques which can be applied to detect and recover data hidden using each of these methods.

MATERIALS AND METHODS

System overview

The most important reason behind the idea of this system is to resolve the need for programmers to always create a back door in the application that they develop as a temporal solution to many problems. This 'make shift' solution and makes customers assume that all programmers have the ability to hack into their system at any time. As a result, most, if not all customers, prefer to employ trusted programmers to build applications on their behalf. Programmers do not create unsafe applications on purpose. On the contrary, they too would like their application to be safe without the need to build ethical relations with their customers.

In this system, a solution is suggested to resolve this problem; the solution is to hide the data in the Unused Area1+ Unused Area

2 + Image Pages; within an executable file of the same system and then another application to be retracted by the customer himself. When Steganography is used, one needs to know all file formats in order to find a way to hide the information in those files. This technique is not easy because in most situations there are large numbers of the file formats and some of them do not permit the hiding of information in them.

System concept

The concept behind this system can be summarized as the hiding of data or any information in the Unused Area1+ Unused Area 2 + Image Pages; within an executable file, so there is no function or routine (open-file, read, write, and close-file) in the operating system to extract it. An alternative method to this operation would be building the file handling procedure independently from the operating system file handling routines. In this case, we need to replace the existing file handling routines by developing a new function which can perform according to our needs, but while maintaining the same exe.file names.

The advantage of this method does not need any additional functions which can be identified by the analysts, and it can be executed remotely and is suitable for network and internet applications.

The disadvantage of this method is that it needs to be physically installed (cannot be operated remotely). Due to this, we choose to implement the first concept in this paper.

System features

This system has the following features:

1. The hiding operation which involves data in the Unused Area1+ Unused Area 2 + Image Pages; within an EXE file using advanced encryption Standard and a statistical technique which increases the degree of security of the information hiding, is used in the proposed system. Since the data which is embedded inside the EXE file does not embed directly in the EXE file, it will be hidden in the triplex space within the EXE file. Thus the attacker is not able to detect the presence of hidden information.
2. The extraction operation: It is very difficult to extract the hidden information just as it is difficult to detect the presence of hidden information. This is because of three reasons:

(i) The hidden information will be encrypted before hiding it using the AES method; this method uses a very strong 128-bit key that would, in theory, be in a range of a military budget within 30 - 40 years. To get an idea of the current status of the AES, the following example is given; whereby we assume that an attacker with the capability to build or purchase a framework that uses keys at the rate of one billion keys per second. This is at least 1 000 times faster than the fastest personal computer in 2004. Based on this assumption, the attacker will require about 10 000 000 000 000 000 000 000 years to try all possible keys for the weakest version.
(ii) It is impossible for the attacker to guess that there is information hidden inside the EXE file since he cannot guess the real size of both the EXE file and information hiding.
(iii) The information hiding should be decrypted after retracting the information.

3. The cover file can be executed just like any other EXE file after the hiding operation, because the hidden information already hides Unused Area1+ Unused Area 2 + Image Pages and it cannot be manipulated as the exe.file, therefore, the cover file is still natural, works normally and is not effected in any way. For instance if the cover EXE file is WINDOWES XP SETUP after the hiding operation, it will continue working as usual, in other words, the EXE file is installed as part of windows operating system.
4. Virus detection programmers' are not capable of distinguishing between such files and the normal EXE, this is due to the style used in the detection of viruses, which uses programs that avoid additional disclosures. Even though there are many forms of detection methods that can be employed by anti-viruses, they mostly depend on the style of string matching and forms. Many of these do not reveal evidence added to the signature and they only depend on disclosed known viruses. While other methods such as heuristics, depend on the type of expectations and the detection of unknown viruses.

The technique used to distinguish between the added code and the norm uses a series of directives; such as a row or a change in the entry point of the program code. Matching refers to the evidence found in the structure of the program when compared to its form at the point of installing the program and this is a very cardinal aspect in our study, because the addition of data to the triplex space may require some changes. This is done so that the structure taken in the process of concealment does not wind up having any changes in its file composition and structure. This is what makes the EXE file undetectable by Unit-Virus.

Triplex space structure

The programming used in implementing the system is known as Java. The PE file layout is shown in Figure 1, there are three free spaces in the PE file layout which were used for this system (Zaidan et al., 2009), and these vacant spaces were selected for hiding the steganography because:

1. The size of the free area 1 is same for all such files.
2. The size of the free area 2 is different for all such files.
3. The size of the image pages is different for all such files.

TESTING OF THE RESULTS

The two essential approaches for identifying test cases are referred to as functional and structured testing. Both of them have a number of distinct test case identification methods which are typically known as testing methods. Functional testing, work's on the assumption that every program can be considered to be a function that maps values from its input domain to values in its output range (Function, domain and range). This perception is generally used in engineering (Zaidan et al., 2009). The advantages of functional test cases are: The independence with which the software is implemented; such that in the event, the implementation changes, the test cases remain useful and test case development can take place in parallel with the implementation, by so doing, the overall research development interval is lessened. However, functional test cases repeatedly exhibit two problems: There can be significant redundancies among test cases, and this is compounded by the possibility of gaps in untested software. This is as shown in Figure 2

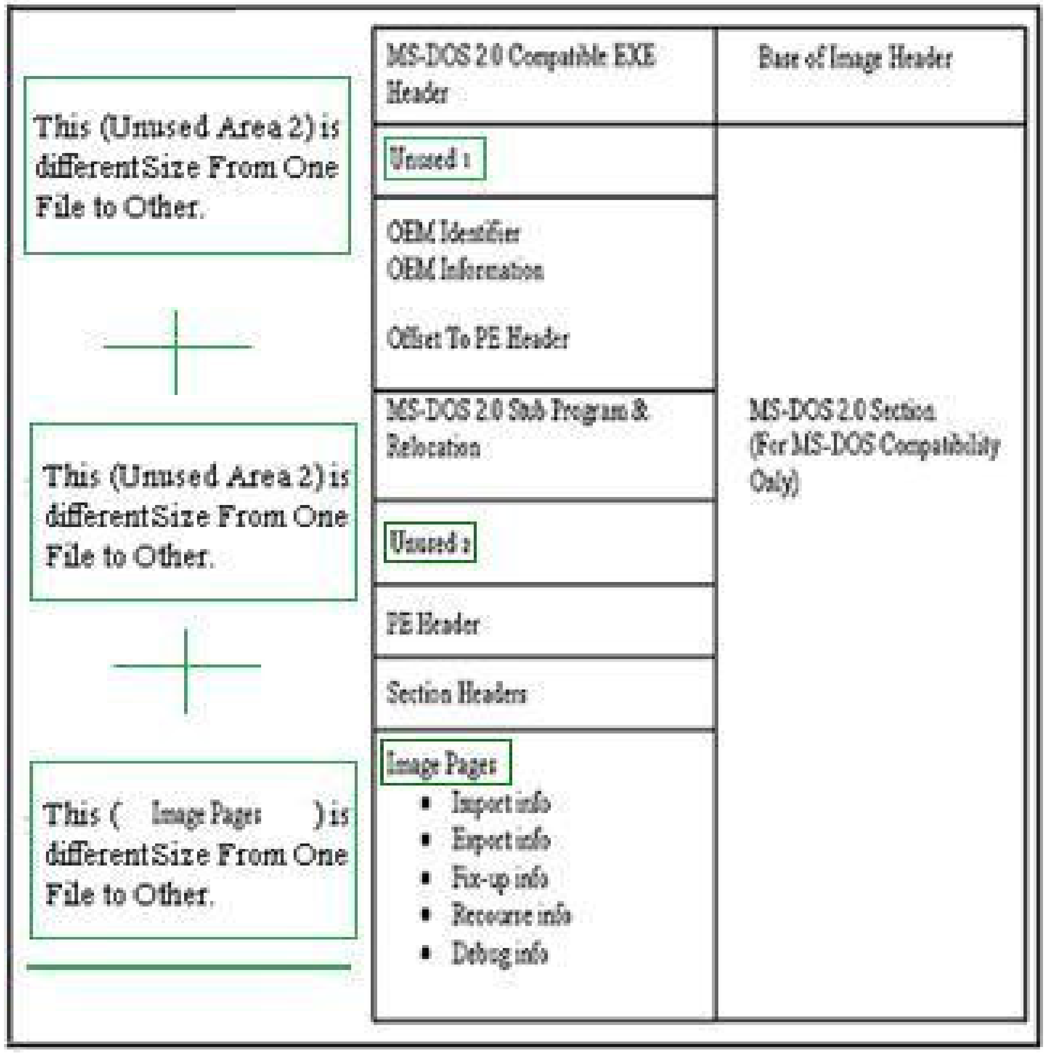

This (Unused Area 2) is different Size From One File to Other.
This (Unused Area 2) is different Size From One File to Other.
This (Image Pages) is different Size From One File to Other.
MS-DOS 2.0 Compatible EXE Header
Unused 1
OEM Identifier
OEM Information
Offset To PE Header
MS-DOS 2.0 Stub Program & Relocation
Unused 2
PE Header
Section Headers
Image Pages
Import info
Export info
Fix-up info
Resource info
Debug info
Base of Image Header
MS-DOS 2.0 Section (For MS-DOS Compatibility Only)

Figure 1. Typical 32-bit portable exe.file layout.

(Naji et al., 2009).

When systems are regarded as being "black boxes", test cases are produced and executed from the specification of the required functionality at defined interfaces; this means that the function of the black box is understood only in terms of its inputs and outputs as shown in Figure 3.

It is not necessary for the code being tested to be seen. In some situations code will not be provided in source code form, yet it can still generate useful test in its absence. In addition to this, the one writing the test cases does not require an understanding of the implementation (Muhammad et al., 2009). Black-box testing still has some significant advantages:

1. Since the test cases are not dependent on the implementation, they can be written simultaneously with or prior to the implementation. What's more, black-box test cases that are properly done do not need to be modified, even if the implementation is entirely rewritten.
2. Creating black-box testing enables the programmer to give careful thought to the specification and its implications. Scores of specification errors are uncovered this way.

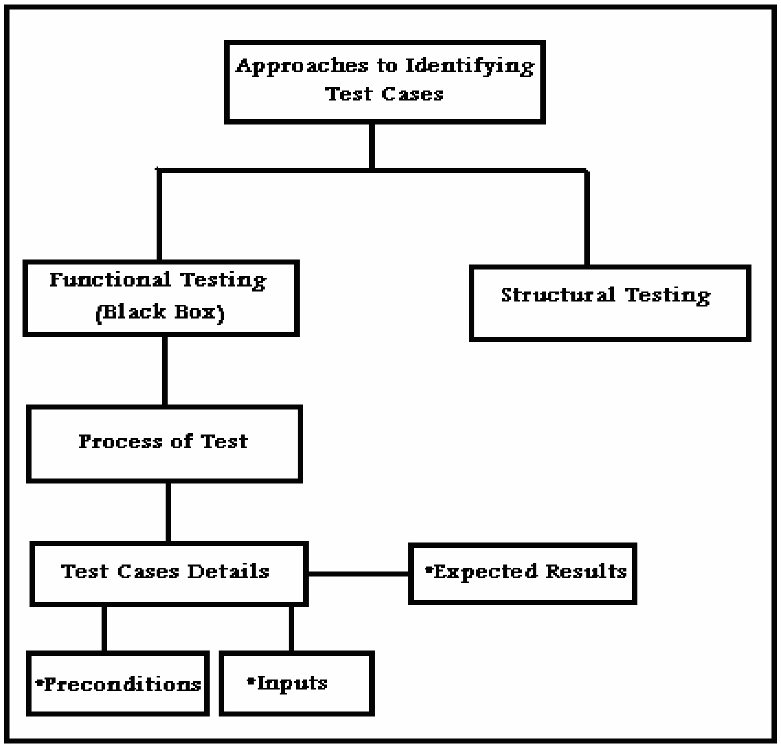


Figure 2. Approaches to Identifying test cases.

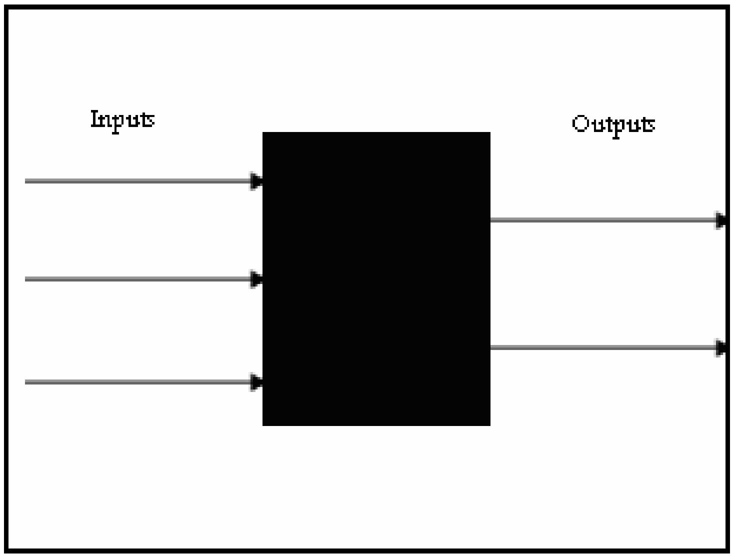


Figure 3. Black Box.

The drawback associated with black box testing is that; its coverage may not be high enough, because it has to work without the implementation. However, it serves as a starting point for writing test cases when the functional approach to test case identification is used; the only information that is available is the specification of the software.

Process of the test

Test case one

In this stage, comparison is made between the cover file size prior to and after the hiding operation: Table 2 shows different sizes of the cover with different types of the exe.files and different sizes of information for each type of multimedia file (text, video, audio or image).

Test case two

In this stage, the test for the usage of exe.files after the hiding operation is done: Four pictures are selected to be used as the cover (exe.Files). The usage after the hiding operation and these pictures is demarcated into:

i) First picture of text.
ii) Second picture of image.
iii) Third picture of video.
iv) Fourth picture of audio.

Test case three

In this stage, the testing for Scanning Results (undetectable by antivirus software) is done: Four pictures are selected to be used as the cover (exe files) which are undetectable by antivirus software after the hiding operation and these pictures are demarcated into:

i) First picture of text.
ii) Second picture of image.
iii) Third picture of video.
iv) Fourth picture for audio.

Test cases details

Test cases are worked out prior to the test being executed, hence; inputs and expected results are known as preconditions. The definition of software installation required for the test is ‘Preconditions’, the definition inputs required for the test is ‘Inputs’ and the definition of predictable results for outputs is ‘Except Results’.

Preconditions

1. Installation (Microsoft Windows XP and above).
2. Installation (Java Net beans).
3. Installation (Microsoft Office Word Document 2003 or 2007).
4. Installation (Software Antivirus).
5. Installation (Real Player Programme).
6. Installation (Jet Audio Programme).
7. Installation (ACDSEE Programme).
8. System application for this research.

Inputs

The system has two types of inputs as shown in the Table 1.

1. Inputs for cover (exe.Files): Four types of the cover (exe.Files) of different size.
2. Inputs for information hidden:

i) Four texts of different size.
ii) Four images of different size.
iii) Four videos of different size.
iv) Four audios of different size.

Expected results

1. Secure cover (exe.Files).
2. The hidden information can be of any type of multimedia file.
3. These are used as covers (exe.Files) after the hiding operation.
4. Antivirus software is unable to detect these covers (exe.Files) after the hiding operation.

Explain the test cases

Test case one

This test case is shown in Table 2 of the cover files and information hidden before and after the hiding operation of all types of multimedia files (text, image, audio and video), which are related to this system. These covers (exe.Files) are selected because they are secure and there are no limitations on the hidden file size.
From the test case in Table 2, one can conclude that:

1. The simulation shows that the size of the hidden data does depend on the size of the Unused Area1+ Unused Area 2 + Image Pages; within the cover.
2. The attacker can not attack the information hiding, because he cannot guess the exe.file size in that the exe.file size does not maintain a constant size, as seen in Table 2 which shows different sizes of the same type of exe.files, for instance cover file number 4 has three sizes

Table 1. Inputs for test cases.

Name of inputs	Type of inputs	Size of inputs/bytes
Cover 1	VM Ware player setup	4,027,802
Cover 2	SSH	532,480
Cover 3	J creater editor setup	22,806,060
	JDK setup	68,830,610
Cover 4	JDK setup	76,445,080
	JDK setup	81,208,728
Text 1	Word document	10,752
Text 2	Word document	10,001
Text 3	Word document	20,100
Text 4	Word document	90,888
Video 1	Real player	200,913
Video 2	Real player	25,333
Video 3	Real player	930,102
Video 4	Real player	3,998,913
Audio 1	Jet audio	160,872
Audio 2	Jet audio	20,332
Audio 3	Jet audio	750,702
Audio 4	Jet audio	2,281,008
Image 1	JPEG	120,440
Image 2	JPEG	150,833
Image 3	JPEG	500,989
Image 4	JPEG	888,431

Table 2. Different Size of the Cover with Different Type of the exe.Files and Different Size for the Information.

Before hide information				After hide information	((size of cover after hide information – size of cover before hide information)/ size of cover before hide information)+100%
Information hidden	**No. of cover**	**Size of IH/bytes**	**Size of cover/ bytes**	**Size of cover/ bytes**	**(%)**
Test 1	1	10,752	4,027,802	4,038,554	0,266
Test 2	2	10,001	532,480	542,481	1,878
Test 3	3	20,100	22,806,060	22,826,160	0,088
Test 4	4	90,888	68,830,616	68,921,504	0,132
Image 1	1	120,440	4,027,802	4,148,242	2,990
Image 2	2	15,383	532,480	547,863	2,888
Image 3	3	500,989	22,806,060	23,307,049	2,196
Image 4	4	888,431	76,445,080	77,333,511	1,162
Audio 1	1	160,872	4,027,802	4,188,674	3,994
Audio 2	2	20,332	532,480	552,812	3,818
Audio 3	3	750,702	22,806,060	23,556,762	3,291
Audio 4	4	2,281,008	81,208,728	83,489,736	2,808
Video 1	1	200,913	4,027,802	4,228,715	4,988

Table 2. Contd.

Video 2	2	25,333	532,480	557,813	4,757
Video 3	3	930,102	22,806,060	23,736,162	4,078
Video 3	4	3,998,913	81,208,728	85,207,641	4,924

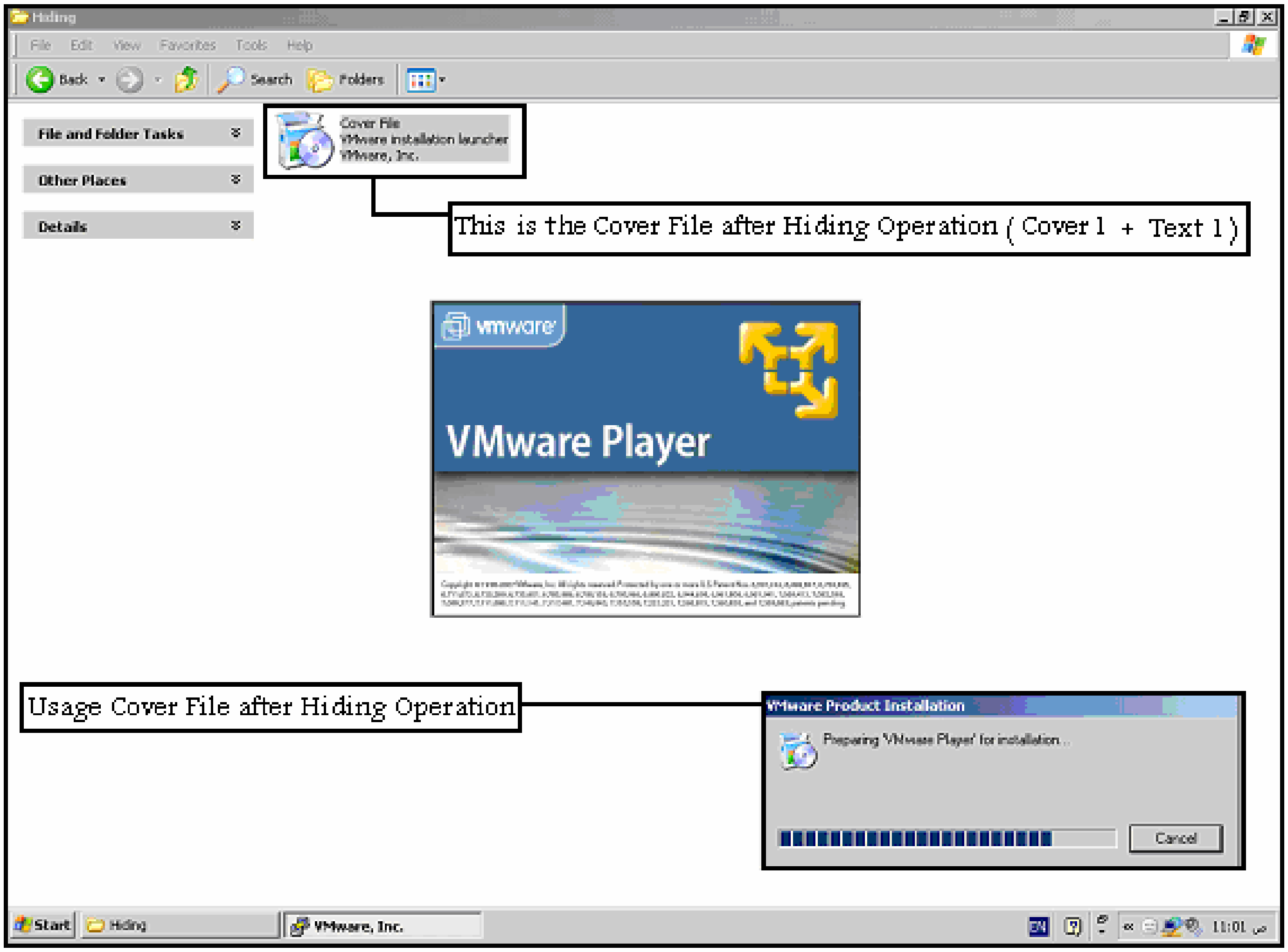

Hiding
File Edit View Favorites Tools Help
Back Search Folders
File and Folder Tasks
Other Places
Details
Cover File
VMware installation launcher
VMware, Inc.
This is the Cover File after Hiding Operation (Cover 1 + Text 1)
vmware
VMware Player
Usage Cover File after Hiding Operation
VMware Product Installation
Preparing 'VMware Player' for installation...
Cancel
Start
Hiding
VMware, Inc.
11:01

Figure 4. After the hiding operation is inside the (hiding folder), the executable file (cover 1) still works.

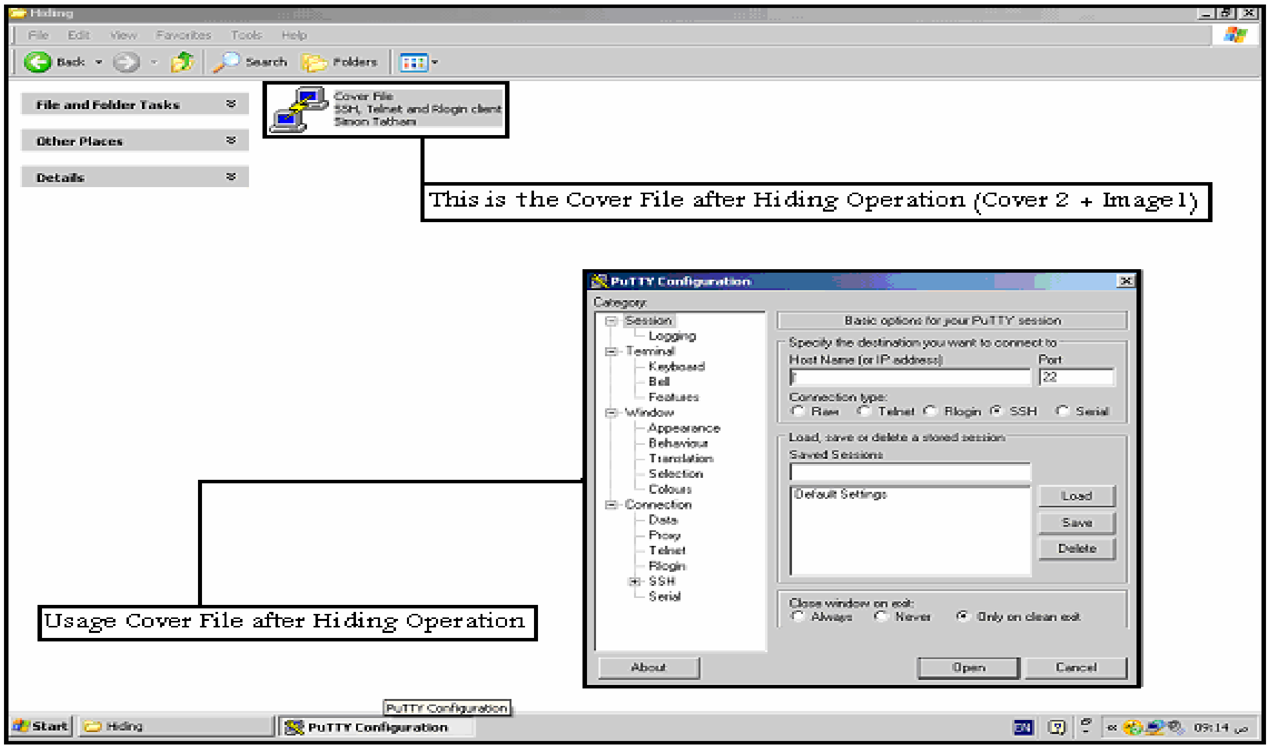
for the same type of the cover file.

Test case two

This test case shows the cover files after the hiding operation of all types of multimedia files (text, image, audio and video) as shown in (Figures 4, 5, 6 and 7) respectively, which are related to this system, these cover (exe.files) were selected because of their ability to be used after the hiding operation.

Test case three

This test case shows the cover files after the hiding operation of all types of multimedia files (text, image, audio and video) as shown in (Figures 8, 9, 10 and 11) respectively, which are related to this system, these covers (exe.Files) were selected because they are

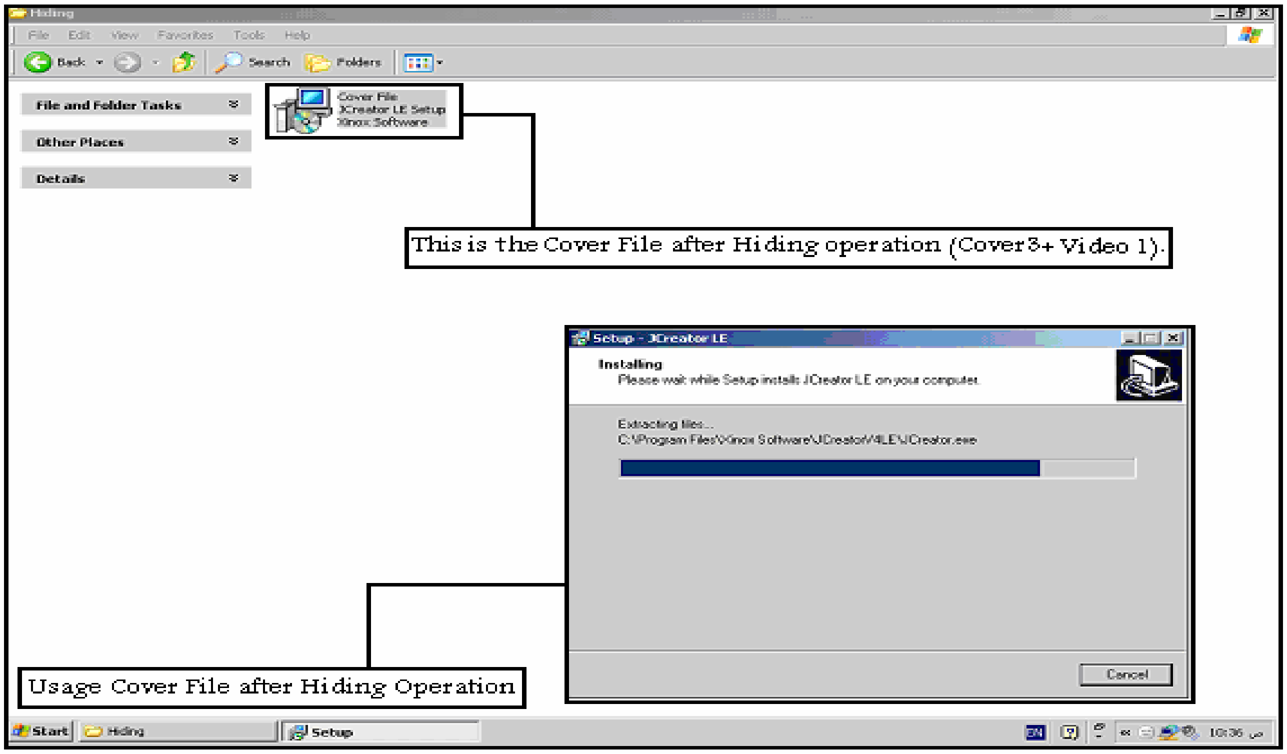


This is the Cover File after Hiding Operation (Cover 2 + Image 1)

Usage Cover File after Hiding Operation

Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Figure 5. After the hiding operation is Inside the (Hiding Folder), the executable file (Cover 2) still works.



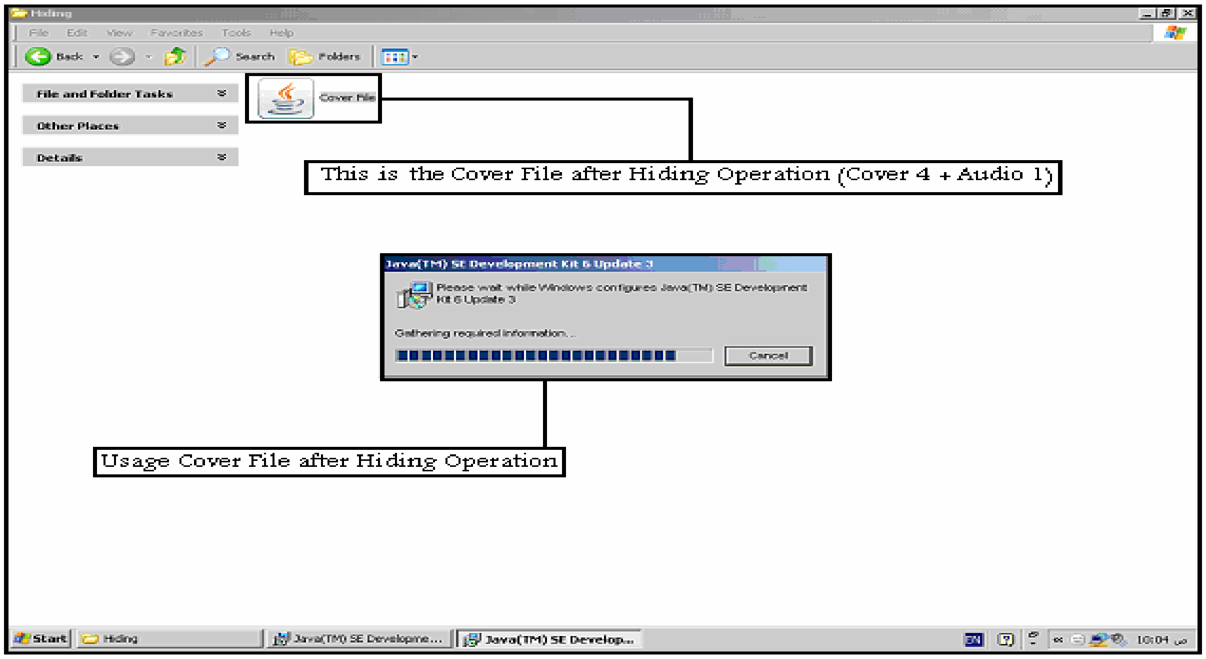
This is the Cover File after Hiding operation (Cover3+ Video 1).

Usage Cover File after Hiding Operation

Installing
Please wait while Setup installs JCreator LE on your computer.

Extracting files...
C:\Program Files\Xinox Software\JCreatorV4LE\JCreator.exe

Figure 6. After the hiding operation is Inside the (hiding folder), the executable file (Cover 3) still works.



This is the Cover File after Hiding Operation (Cover 4 + Audio 1)

Java(TM) SE Development Kit 6 Update 3

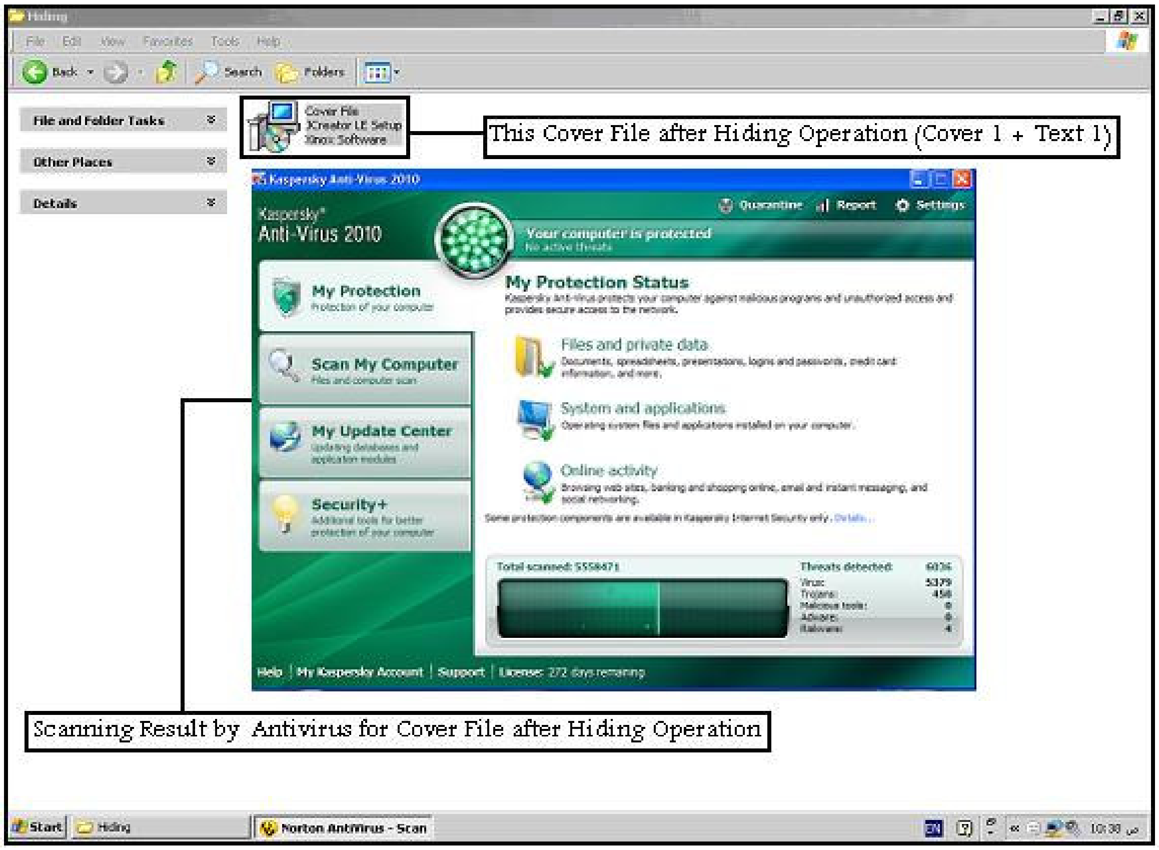
Please wait while Windows configures Java(TM) SE Development Kit 6 Update 3

Gathering required information...

Cancel

Usage Cover File after Hiding Operation

Figure 7. After the hiding operation is inside the (Hiding Folder), the executable file (Cover 4) still works.



This Cover File after Hiding Operation (Cover 1 + Text 1)

Your computer is protected
No active threats

My Protection Status

Kaspersky Anti-Virus protects your computer against malicious programs and unauthorized access and provides secure access to the network.

Files and private data
Documents, spreadsheets, presentations, logins and passwords, credit card information, and more.

System and applications
Operating system files and applications installed on your computer.

Online activity
Browsing web sites, banking and shopping online, email and instant messaging, and social networking.

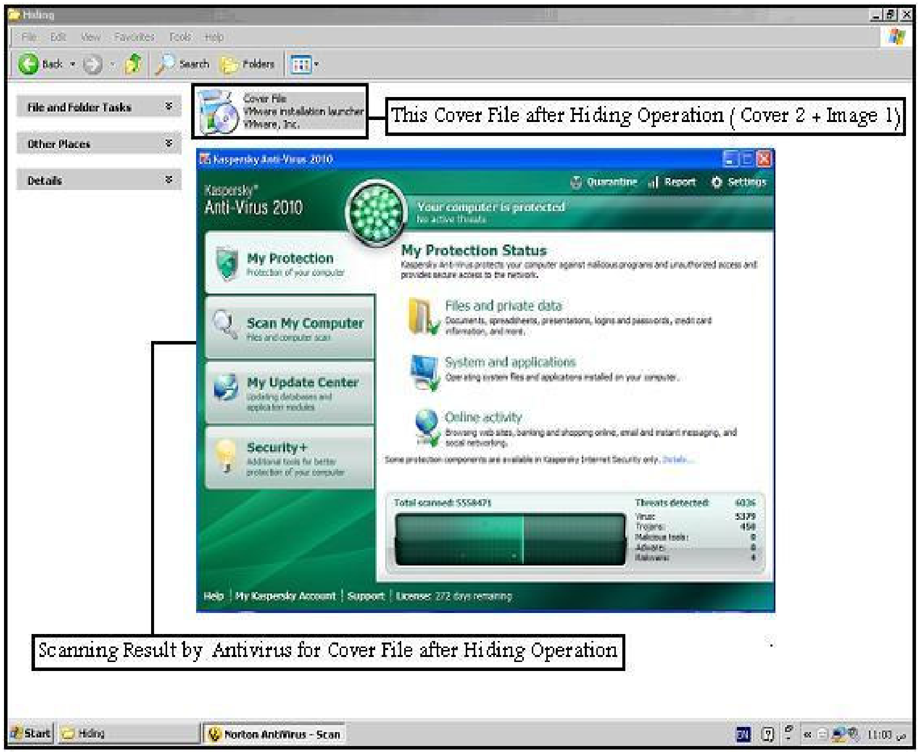
Some protection components are available in Kaspersky Internet Security only. Details...

```
Total scanned: 5558471      Threats detected:   6036
                            Virus:              5379
                            Trojans:             458
                            Malicious tools:       0
                            Adware:                0
                            Riskware:              4
```

License: 272 days remaining

Scanning Result by Antivirus for Cover File after Hiding Operation

Figure 8. Shows that the executable file (Cover 1) inside (Hiding Folder) immune to anti-virus program.


Cover File
VMware installation launcher
VMware, Inc.

This Cover File after Hiding Operation (Cover 2 + Image 1)

Kaspersky Anti-Virus 2010

Your computer is protected
No active threats

My Protection Status
Kaspersky Anti-Virus protects your computer against malicious programs and unauthorized access and provides secure access to the network.

Files and private data
Documents, spreadsheets, presentations, logins and passwords, credit card information, and more.

System and applications
Operating system files and applications installed on your computer.

Online activity
Browsing web sites, banking and shopping online, email and instant messaging, and social networking.

Some protection components are available in Kaspersky Internet Security only. Details...

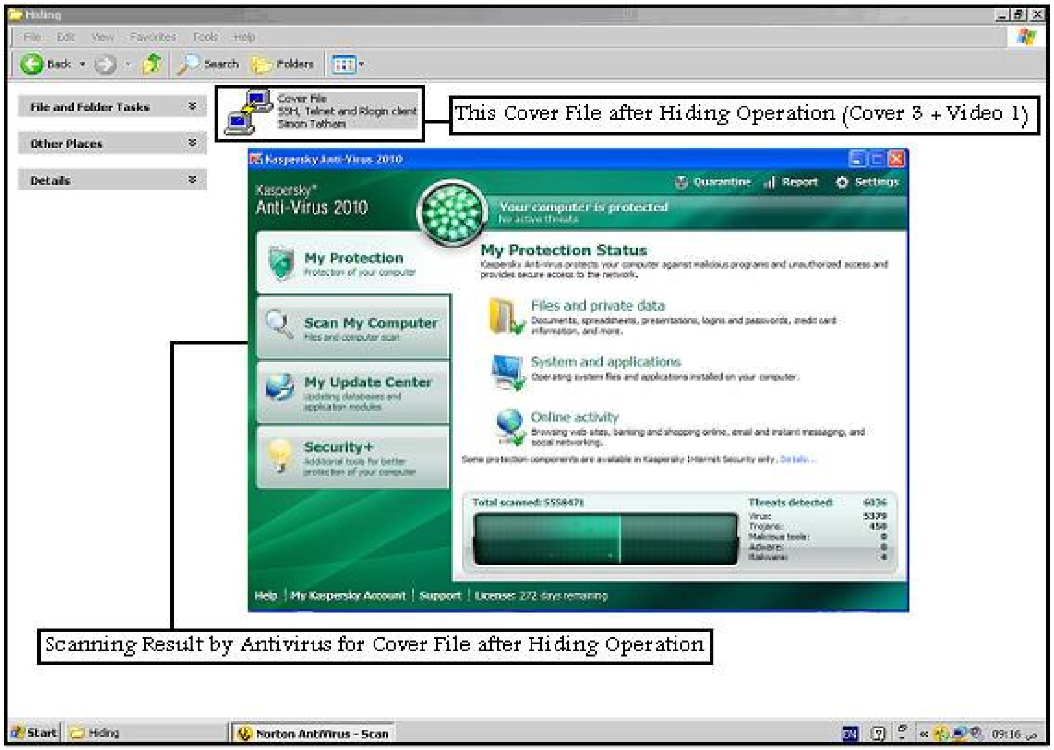
Total scanned: 5558471

Threats detected:	6036
Virus:	5379
Trojans:	458
Malicious tools:	0
Adware:	0
Riskware:	4

Help | My Kaspersky Account | Support | License: 272 days remaining

Scanning Result by Antivirus for Cover File after Hiding Operation

Figure 9. Shows that the executable file (cover 2) inside (hiding folder) immune to anti-virus program.


Cover File
SSH, Telnet and Rlogin client
Simon Tatham

This Cover File after Hiding Operation (Cover 3 + Video 1)

Kaspersky Anti-Virus 2010

Your computer is protected
No active threats

My Protection Status
Kaspersky Anti-Virus protects your computer against malicious programs and unauthorized access and provides secure access to the network.

Files and private data
Documents, spreadsheets, presentations, logins and passwords, credit card information, and more.

System and applications
Operating system files and applications installed on your computer.

Online activity
Browsing web sites, banking and shopping online, email and instant messaging, and social networking.

Some protection components are available in Kaspersky Internet Security only. Details...

Total scanned: 5558471

Threats detected:	6036
Virus:	5379
Trojans:	458
Malicious tools:	0
Adware:	0
Riskware:	4

Help | My Kaspersky Account | Support | License: 272 days remaining

Scanning Result by Antivirus for Cover File after Hiding Operation

Figure 10. Shows that the executable (cover 3) file inside (hiding folder) immune by anti-virus program.

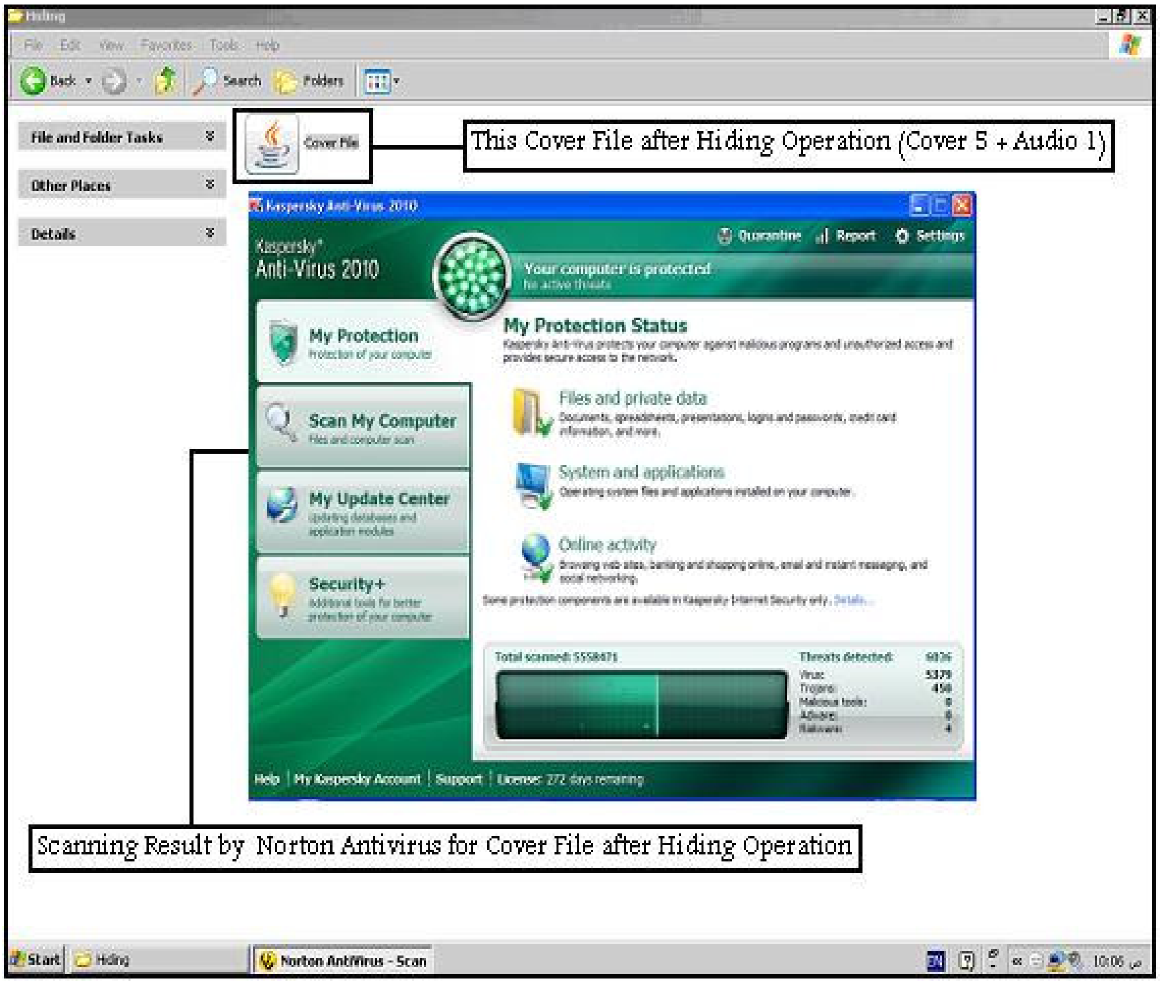

This Cover File after Hiding Operation (Cover 5 + Audio 1)
Cover File
Scanning Result by Norton Antivirus for Cover File after Hiding Operation
Kaspersky Anti-Virus 2010
Your computer is protected
My Protection
Scan My Computer
My Update Center
Security+
My Protection Status
Files and private data
System and applications
Online activity
Norton AntiVirus - Scan

Figure 11. Shows that the executable (cover 5) file inside (hiding folder) immune to anti-virus program.

undetectable by antivirus software after the hiding operation.

DISCUSSION

1. The size of the hidden data does depend on the size of the Unused Area1+ Unused Area 2 + Image Pages; within the cover file which is equal to 28% of the size of exe.file before the hiding process.
2. The executable files still works after it is used as a cover for embedding data.
3. The executable file is undetectable by the Norton antivirus software after the hiding operation.
4. The sizes of cover files used in tests are 532,480 - 81,208,728 Byte, the size of the information hiding using text (10,001 Byte - 90,888 Byte), Image (15,383 –888,431 Byte), audio (20,332 - 2,281,008 Byte) and video (25,333 - 3,998,913 Byte). From the test, it was found that the hiding method makes it possible for the cover and the message to be independent. In the event that the size of the cover exe.Files is upgraded, the security also increases. Also when the information hidden inside the cover is less than the cover file size, the information hiding is even more secure.

Conclusion

In this paper, a new approach for high secure and rate data hidden within triplex space for non multimedia file steganography has been presented. The basis of this method is the use of an executable file as a cover file that hides the information. The new approach's successes are based on the hiding, encryption, extraction, and decryption functions which do not negatively affect the functionality of the EXE file. This framework overcomes the limitation of the steganographic approach by the usage of the largest cover file size from among the non multimedia files which is the EXE. By using the EXE file, we have the flexibility of making a computation that involves steganography and cryptography which makes it possible to achieve a higher security level for the system or using the triplex space for the EXE file to hide huge amounts of data. To mitigate security issues, the authors have chosen the AES Algorithm method to guarantee the protection of data even if the attacker somehow gets hold of the data. Hence one of important conclusions is that since most antivirus systems do not allow direct writing into executable files, the approach for the proposed system is able to prevent the hidden information from being observed by these systems and at the same time, the cover file can also be executed normally after the hiding operation. In other words, the cover file still behaves naturally, that is, working normally and is not affected in any way.

ACKNOWLEDGMENTS

We wish to thank all the researchers who contributed to this project. Also we would like to acknowledge and thanks the researchers in MMU for their unlimited support and this project was funded by the University of Malay, IPPP, Project No: P009/2008A.

REFERENCES

Ahmed A, Mohamed M, Kiah ML, Zaidan AA, Zaidan BB (2010). A Novel Embedding Method to Increase Capacity of LSB Audio Steganography Using Noise Gate Software Logic Algorithm ', Journal of Applied Sciences, Vol.10, Issue 1, ISSN: 1812-5654, pp. 59-64.

Anckaert BB, De Sutter D, Chanet and. De Bosschere K (2005). 'Steganography for executable and code transformation signatures '. Proceedings of the 7th Information Security and Cryptology May 24, Springer Berlin, Heidelberg, pp. 425-439. http://www.springerlink.com /content/vbxjdapj9g 25 agel.

Artz D (2001). Digital steganography for hiding data within data, Internet Computing, IEEE, 5(3): 75-80. URL:http://ieeexplore.ieee.org/xpl/ freeabs_all.jsp?&arnumber=935180.

Bassia PP I, Nikolaidis N (2001). 'Robust audio watermarking in the time domain', Multimedia, IEEE Transactions on, 3(2): 232 -241. URL:http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=923822

Bender WDGN, Morimoto AL (1996). Techniques for data hiding, IBM Systems J., 35(3-4): 313-336. URL:http://portal.acm.org/citation.cfm?id=243522&dl=GUIDE&coll=GUIDE&CFID=56553217&CFTOKEN=80925059.

Cvejic NST (2002). 'Increasing the capacity of LSB-based audio steganography', Multimedia Signal Processing, IEEE Worksho, pp. 336-338URL:http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1203314.

Cvejic NST (2004). 'Reduced distortion bit-modification for LSB audio steganography'. Proc. Signal Processing. Proceedings. ICSP '04. 2004 7th International Conference2004 pp. 2318- 2321, URL:http://www.mediateam.oulu.fi/publications/pdf/550.pdf.

Cvejic NST (2005). Increasing Robustness of LSB Audio Steganography by Reduced Distortion LSB Coding', journal of universal computer science, 11(1): 56- 65, URL:ftp.math.utah.edu/ pub//tex/bib/cryptography2000.ps.gz

El-Khalil R, Keromytis AD (2004). Hiding information in program binaries'. Proceedings of the 6th International Conference on Information and Communications Security, Oct. 27- 29, Springer Berlin, Heidelberg,pp: 187199.http://cat.inist.fr/?aModele=afficheN&cpsidt=16334236.

Lee YKC, Chen LH (2000). High capacity image steganographic model, Vision, Image and Signal Processing, IEE Proceedings, 147(3): 288-294. URL:http://ieeexplore.ieee.org/xpl/freeabs_all.jsp? Arnumber =852312.

Muhammad IAS, Zaidan MA, Zaidan AA, Zaidan BB (2009). Student record retrieval system using knowledge sharing. Int. J. Comput. Sci. Network Secur. 9: 97-106. .http://paper.ijcsns.org/07_book/ 200906/20090614. Pdf

Naji AW, Zaidan AA, Zaidan BB, Shihab A, Khalifa OO (2009). Novel approach of hidden data in the unused area 2 within exe files using computation between cryptography and steganography. Int. J. Comput. Sci. Network Secur. 9: 294-300. http://paper.ijcsns.org/ 07_book/200905/20090539. Pdf.

Noto M (2001). MP3Stego: Hiding Text in MP3 Files, SANS Institute,URL:http://www.sans.org/reading_room/whitepapers/stenganography/mp3stego_hiding_text_in_mp3_files_550?show=550.php&cat=stenganograph.

Ross JA (1998). On the Limits of Steganography, IEEE Journal of Selected Areas in Communications, 16(4): 474-481. URL:http:// www.petitcolas.net/fabien/publications/jsac98-limsteg.pdf.

Yeh CH, Kuo CJ (1999). Digital watermarking through quasi m-arrays , Signal Processing Systems. SiPS 99. IEEE Workshop oni, pp. 456-461. URL:http://ieeexplore.ieee.org/xpl/freeabs_all.jsp? Arnumber =822351.

Zaidan AA, Zaidan BB, Abdulrazzaq MM, Raji RZ, Mohammed SM (2009). Implementation stage for high securing cover-file of hidden data using computation between cryptography and steganography. Int. Assoc. Comput. Sci. Inform. Technol., 20, Session 6: 482-489. http://WWW.IACSIT.ORG and Www.WordAcademicPress.com.