



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DETECTION AND MITIGATION OF ANTI-FORENSICS USING FORENSIC TOOLS

by

Emre Caglar Hosgor

December 2018

Thesis Advisor:
Second Reader:

Neil C. Rowe
Glenn R. Cook

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2018	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE DETECTION AND MITIGATION OF ANTI-FORENSICS USING FORENSIC TOOLS			5. FUNDING NUMBERS	
6. AUTHOR(S) Emre Caglar Hosgor				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Although information technology has improved our living standards, it has also provided criminals new ways to commit crime. Digital crime includes identity theft, online piracy, hacking, and terrorism. For combating digital crime, new techniques and tools emerge frequently in digital forensics. On the opposite side, cyber-criminals develop counter-techniques called anti-forensics, which aim to disrupt or manipulate forensic analysis of crime. This thesis investigated the effectiveness of some representative anti-forensic tools for data hiding, artifact wiping, and trail obfuscation. We found they varied considerably in effectiveness and a variety of countermeasures can be used against them.				
14. SUBJECT TERMS computer forensics, anti-forensics, computer forensic tools			15. NUMBER OF PAGES 83	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**DETECTION AND MITIGATION OF ANTI-FORENSICS USING FORENSIC
TOOLS**

Emre Caglar Hosgor
Captain, Army, Turkey
BS, Turkish Military Academy, 2006

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
December 2018**

Approved by: Neil C. Rowe
Advisor

Glenn R. Cook
Second Reader

Dan C. Boger
Chair, Department of Information Sciences

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Although information technology has improved our living standards, it has also provided criminals new ways to commit crime. Digital crime includes identity theft, online piracy, hacking, and terrorism. For combating digital crime, new techniques and tools emerge frequently in digital forensics. On the opposite side, cyber-criminals develop counter-techniques called anti-forensics, which aim to disrupt or manipulate forensic analysis of crime. This thesis investigated the effectiveness of some representative anti-forensic tools for data hiding, artifact wiping, and trail obfuscation. We found they varied considerably in effectiveness and a variety of countermeasures can be used against them.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	LITERATURE REVIEW	3
A.	ANTI-FORENSICS	3
B.	CATEGORIZATION OF ANTI-FORENSIC TECHNIQUES.....	4
C.	DESCRIPTION OF ANTI-FORENSIC CATEGORIES	6
1.	Data Hiding.....	6
2.	Artifact Wiping	9
3.	Trail Obfuscation	10
4.	Attacks against Digital Forensic Tools and Processes	10
III.	ANALYSIS OF ANTI-FORENSIC TOOLS	13
A.	DATA HIDING	14
1.	File System Data Hiding Tools and Techniques	14
2.	Network-Communications Data Hiding	17
3.	Using Encryption for Data Hiding	20
4.	Steganography and Example Implementation	21
B.	ARTIFACT WIPING TECHNIQUES	24
1.	File Wiping	24
2.	Generic-Data Wiping and Registry Wiping	25
3.	Metadata Wiping	26
C.	TRAIL OBFUSCATION.....	26
D.	ATTACKS AGAINST FORENSIC TOOLS AND PROCESSES	26
IV.	MITIGATION OF ANTI-FORENSICS AND RECOMMENDATIONS	27
A.	DATA HIDING	27
1.	Detection of File System Data Hiding	27
2.	Detection of Network Communication Data Hiding with Stunnel	28
3.	Detection and Mitigation Techniques against Encryption Usage for Data Hiding	28
4.	Detection of Steganography	29
B.	ARTIFACT WIPING	29
C.	MITIGATION TECHNIQUES AGAINST TRAIL OBFUSCATION AND ATTACK AGAINST FORENSIC TOOLS.....	30
D.	SUMMARY	30

APPENDIX A. ANTI-FORENSIC TOOLS CONSIDERED	33
APPENDIX B. INSTRUCTIONS FOR ANTI-FORENSIC TOOLS.....	43
A. BMAP TOOL INSTALLATION, CONFIGURATION PROCESS AND USAGE EXAMPLE.....	43
B. STUNNEL TOOL INSTALLATION, CONFIGURATION AND USAGE.....	44
C. IMAGE STEGANOGRAPHY TOOL SOURCE CODE.....	45
D. PYTHON SCRIPT FOR SHANNON ENTROPY CALCULATIONS	53
E. METADATA WIPING AND TRAIL OBFUSCATION EXPERIMENTS	55
F. EVENT LOG MANIPULATION.....	57
LIST OF REFERENCES	59
INITIAL DISTRIBUTION LIST	63

LIST OF FIGURES

Figure 1.	Digital Forensic Examination Stages. Source: Yusoff et al. (2011).	4
Figure 2.	FAT32 and EXT3 Strings Command Output	15
Figure 3.	Testing Alternate Data Streams	17
Figure 4.	Stunnel Topology Used for Experiments.....	17
Figure 5.	Stunnel Private Key and Certificate and Certificate-Authority Signed Certificate.....	19
Figure 6.	The Stunnel Configuration for Improving Security	19
Figure 7.	Forensic Analysis of VeraCrypt Volumes with TSK Autopsy and FTK Imager.....	21
Figure 8.	Data Hiding and Retrieval with a Simple Image Steganography Tool.....	22
Figure 9.	Histogram of Base Image in Test.....	23
Figure 10.	Histogram of Steganography Image in Test	24
Figure 11.	Output of the Shred Tool on a Wiped File.....	25
Figure 12.	Test File Timestamps.....	56
Figure 13.	Result of Timestamp Change with timestomp.....	56

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Anti-forensic Tools	13
Table 2.	Metadata Files in NTFS. Adapted from Huebner et al. (2006).	16

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
BEViewer	Bulk Extractor Viewer
BIOS	Basic Input/output System
CA	Certification Authority
CFT	Computer Forensic Tools
CPU	Central Processing Unit
DBAN	Darik's Boot and Nuke
DES	Data Encryption Standard
EXT2	2nd Extended File system
EXT3	3rd Extended File system
EXT4	4th Extended File system
FAT32	File Allocation Table 32 bits
FTP	File Transfer Protocol
GB	Giga Bytes
GPO	Group Policy Objects
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
IRC	Internet Relay Chat
LSB	Least Significant Bit
MB	Mega Bytes
NAT	Network Address Translation
NTFS	New Technology File System
PGP	Pretty Good Privacy
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
POP	Post Office Protocol
RSA	Rivest, Shamir & Adleman

TCP	Transport Control Protocol
TSK	The Sleuth Kit
VPN	Virtual Private Network

EXECUTIVE SUMMARY

New techniques and detection mechanisms emerge often in digital forensics. On the opposite side, cyber criminals develop new techniques against digital forensics, as well (Lillis, Becker, O’Sullivan, & Scanlon, 2016). The collection of those techniques are called as anti-forensics. Anti-forensics became a phenomenon after 2005. Initial techniques were very effective against forensic investigation. However, now we have detection and mitigation methods for individual anti-forensic techniques. This research examines some popular detection and mitigation techniques against anti-forensics.

Garfinkel (2007) provided a well-accepted taxonomy for the anti-forensics: data hiding, artifact wiping, trail obfuscation, and attacks against forensic tools and processes. This research focused on anti-forensic tools. A tool can have multiple purposes, but we experimented with the most-used purpose of the anti-forensic tool. The most effective and popular tool category is encryption since when a file is encrypted, forensic analysis cannot be conducted. Steganography is also popular, a data hiding technique.

In this research, we provide detection or mitigation methods for each subcategory of anti-forensics. Some tool detection depends on recovering installation files, configuration information, and the use of specific “C” libraries. File streams can be detectable with PowerShell scripts. Network covert-channel detection requires retrieval of key data from the target system. Steganography detection depends on statistical analysis.

Trail obfuscation requires an exploit, so keeping systems up-to-date and detecting an exploit is a viable mitigation method. When cyber-criminals use zip bombs or packers against forensic tools, we can stop them by using FTK Imager, TSK-Autopsy, or Magnet Forensics AXIOM.

References

- Garfinkel, S., & Shelat, A. (2003). Remembrance of data passed: A study of disk sanitization practices. *IEEE Security & Privacy*, 99(1), 17–27.
- Lillis, D., Becker, B., O’Sullivan, T., & Scanlon, M. (2016). Current challenges and future research areas for digital forensic investigation. arXiv preprint 1604.03850.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis is dedicated to my wife, Demet, for her support and understanding. I would like to thank my advisors, Dr. Neil Rowe and Mr. Glenn Cook, for their guidance and wisdom.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Since the introduction of the first computers, information technology and digital devices have played significant roles in our daily lives. Cyber criminals and attackers have also been using information technology and digital devices for malicious purposes (Lillis, Becker, O’Sullivan, & Scanlon, 2016). Digital-forensic investigation has become an important part of the criminal investigation because of the increasing criminal usage of the computers (Lillis et al., 2016). Recently digital forensics is facing some challenges due to technical improvements in digital devices and information technology. Increases in storage capacity, improvements on network bandwidth capacity and communication speed, introduction of the device networks in the “Internet of Things,” and continuing development in mobile devices have created challenges to digital-forensic investigation. Those challenges include complexity, diversity, consistency, volume, and time management (Raghavan, 2013).

As the importance of digital forensic increases, anti-forensic techniques emerge (Liu & Brown, 2006). This thesis defines and classifies anti-forensic techniques, and analyzes some representative tools for detecting and mitigating them. We first provide definitions and background on anti-forensic techniques, and describe some easy-to-use and publicly available ones. Chapter III covers implementation of selected anti-forensic techniques in more detail. Chapter IV covers detection and mitigation of those techniques and some tools to do so, and reaches some conclusions.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

A. ANTI-FORENSICS

Foster and Liu (2005) demonstrated early implementations of anti-forensic tools. Liu and Brown (2006) described four aims of anti-forensic techniques:

- Avoiding detection of malicious actions;
- Disrupting the information collection during the forensic investigation;
- Increasing investigation time required; and
- Decreasing the validity of a forensic report or testimony.

Other anti-forensic tools and techniques have started to emerge. The popular “Metasploit” framework now includes the anti-forensic tools Timestomp, Transmogrify, Slacker, and Sam Juicer in the MAFIA (Metasploit Anti Forensic Investigation Arsenal) which implements techniques of Liu and Brown (2006). Another researcher categorized anti-forensics in four categories as “data hiding, artifact wiping, trail obfuscation, and attacks against digital forensic tools” (Garfinkel, 2007, p. 78). This listed aims of anti-forensics as:

- Revealing the existence of the forensic tools;
- Disrupting forensic analysis;
- Attacking forensic examiners, and
- Clearing the evidence of anti-forensic tool’s existence. (Garfinkel, 2007)

Understanding anti-forensic techniques and tools requires understanding the digital-forensic examination process. It includes three main stages: acquisition and preservation, analysis, and presentation (Yusoff, Ismail, & Hassan, 2011). In addition to these, pre-processing and post-processing phases help the investigator to maintain the

integrity of the evidence and results. Figure 1 depicts the stages of a digital-forensic examination. Each stage has feedback to a previous stage.

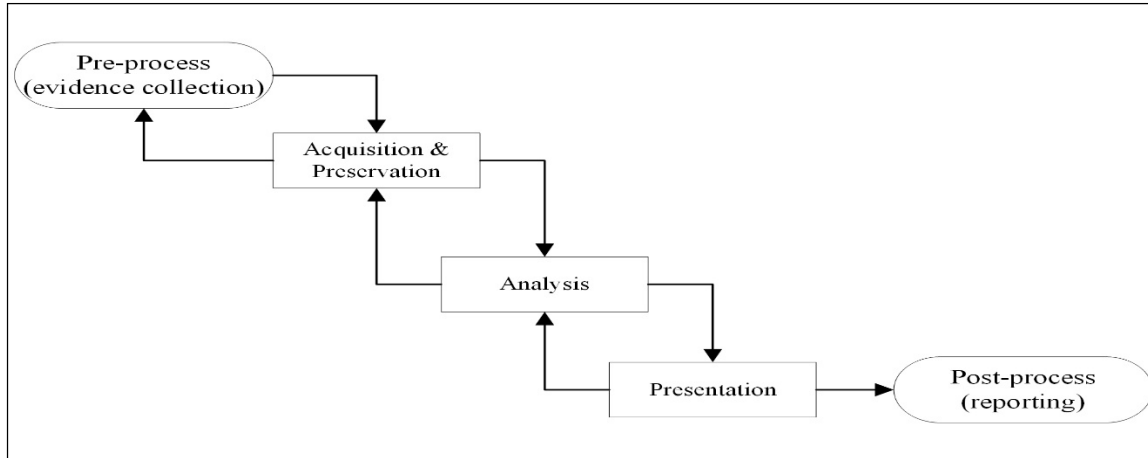


Figure 1. Digital Forensic Examination Stages. Source: Yusoff et al. (2011).

The digital-forensic investigator finds valuable evidence during the analysis stage from the data obtained during the acquisition stage. So anti-forensic tools and techniques target mainly this stage. Some anti-forensic tools target the pre-processing and post-processing stages as well.

B. CATEGORIZATION OF ANTI-FORENSIC TECHNIQUES

Because anti-forensics is an emerging area, new tools are emerging publicly every day. Jahankhani and Beqiri (2010) classify anti-forensics from tool-specific perspective:

- Digital media wiping
- Steganography
- Privacy wipers
- Rootkits
- Homographic attacks

- File signature modification attacks
- Encryption
- Metadata anti-forensics
- Slack space anti-forensics
- Secure digest functions collision generations
- Digital memory anti-forensics
- Misleading evidence
- Packers/binders
- Forensic tools vulnerabilities/exploits
- Resource waste
- Forensic detections
- Anonymous actions
- Anti-forensics in flushable devices

Tool-specific classification of anti-forensics focuses on tool categories. We followed four well-known categories in this research.

- Data hiding
- Artifact wiping
- Trail obfuscation
- Attacks against digital forensic tools and processes

C. DESCRIPTION OF ANTI-FORENSIC CATEGORIES

This project analyzed and correlated anti-forensic tools. An open-source version of a tool was sought where possible. Otherwise, the tool selection of Conlan et al. (2016) was used. Appendix A lists the tools considered.

1. Data Hiding

Data-hiding techniques and tools use file-system, memory, or network capabilities of the operating system for hiding the digital data. Hiding tools, steganography, and encryption are closely related (Bender, Gruhl, Morimoto, & Lu, 2010). However, data hiding is a broader concept and steganography, rootkits, and encryption are specialized hiding techniques. Conlan et al. (2016) suggested subcategories of “data contraception, file system manipulation, hard disk manipulation, memory hiding and network-based hiding.”

a. *File System Data Hiding Tools*

The Microsoft Windows NTFS is a representative file system. NTFS organizes a “volume” or major subpart on a disk with a:

- Boot Sector
- Master File Table
- File System Data
- Master File Table Copy

The Master File Table is the most important portion in an NTFS volume for forensic purposes (Bergel, 2007). When a file data is less than a cluster size, unused space occurs; if a file is larger than a cluster size it, becomes fragmented (Microsoft, 2003). A cyber-criminal can hide data into unused spaces which occur when file-system data does not fill records or when it is fragmented.

The UNIX and Linux operating systems use blocks for file-system structure, and unused or slack space occurs at the Data Block portion of the EXT4 file-system (“Ext4,” n.d.). Slack space provides a place for data hiding in both the Microsoft and UNIX/Linux

file-system structures. Hiding data into slack or unused space is more effective in a Windows environment (Huebner, Bern, & Wee, 2006). Huebner et al. (2006) suggests that a successful data hiding should (1) hide from standard operating-system utilities, (2) have a low chance of being overwritten, (3) hide from the user interface, and (4) store a good amount of data. Slack-space data-hiding techniques meet those characteristics. MAFIA includes a slack-space hiding tool, Slacker, within the Metasploit framework. Slacker was first published in the Metasploit platform in 2006.

b. Memory Data Hiding (Live Hiding) Tools

“Live hiding” tools hide data in main memory (Swanson, Stoller, & Carter, 1998). Main memory is volatile. Nevertheless, retrieval of the data can be done as long as there is electricity to keep memory data on the memory cells. Retrieval techniques are both hardware and software-based, and they are rather easy to implement because anti-forensic tools in the main memory do not try to hide themselves very much (Burdach, 2006).

c. Network-Based Hiding Tools

Network-based data-hiding tools hide data in one of the layers provided in the Internet Protocol Stack model. Techniques are covert channeling, protocol bending, and packet crafting (Bergel, 2007).

Wrapping tools are a good example for network-based anti-forensic tools. They are common and easy to implement. The UNIX Stunnel tool is a well-known network-level SSL wrapper tool (Trojnara, 2016). It wraps unencrypted communication into a SSL tunnel. This tool was developed for providing a secure communication channel for insecure TCP/IP protocols. However, attackers can hide data using it.

Terminal emulators are other network-based anti-forensic tools. However, terminal emulators require administrator/root privileges for an installation on a client machine. Common network emulators that can hide data are AbsoluteTelnet, Indigo Terminal Emulator, and SecureCRT (Conlan, et al., 2016).

A Virtual Private Network (VPN) provides security for private networks over an insecure channel. VPN protocols use encryption for securing data. An attacker can

establish a client-server VPN for exfiltration of sensitive data or data hiding. Encapsulating VPN packets in another protocol is an alternative method for data hiding. A good example of VPN encapsulation is encapsulating VPN packets into an IP datagram, keeping the header information unchanged.

d. Encryption Techniques

A basic definition for the encryption is transforming data into a secret code (Beal, n.d.). Modern encryption algorithms can be synchronous or asynchronous. Encryption provides confidentiality and ways to hide data (Boneh, Sahai, & Waters, 2011). Common encryption targets are files, disks, email, file-systems, applications, and data in transit (Conlan et al., 2016). Whole-disk and file encryption is the focus of this thesis. Encryption disrupts the initial acquisition phase of the digital forensics examination so the examiner cannot complete the following phases.

e. Steganography Techniques

Steganography is techniques to hide secret information in image, video, audio, or text files so that the information cannot be detectable by a naked eye (Mishra et al., 2014). Distortion and spread-spectrum techniques are examples of audio steganography, and substitution techniques are examples of image steganography (Singh & Mahajan, 2016). In all steganography methods, encryption can be used too to provide extra protection against steganalysis. Comparing the aims of steganography and encryption, encryption puts content of the message in an undecipherable form, and steganography hides the message existence as well. Some example stenographic techniques are:

- Substitution. This is the most common and easiest technique. An attacker uses redundant places to hide the secret message such as the least significant bits.
- The discrete cosine transform on audio where one bit is modulated.
- The spread-spectrum technique on audio. The attacker spreads the secret over a wide spectrum and then modulates it into the carrier signal.

- Distortion techniques. The information is encoded in the signal by distorting the cover. The difference between the original cover and the distorted one gives the secret. (Singh & Mahajan, 2016).

f. **Rootkits**

Rootkits are specialized code sectors that hide in the operating-system kernel (Hoglund & Butler, 2006). Rootkits are a type of malicious software that runs at the inner levels of an operating system. Cyber criminals use rootkits not only hiding data, but also for logging the network activity, storing keystrokes, process hiding, and controlling registry entries (Sparks & Butler, 2005).

2. Artifact Wiping

Artifact wiping is an effective method for destroying digital evidence. Garfinkel (2007) and Conlan et al. (2016) identified these methods:

- Disk wiping erases data from a disk (hard or solid-state) securely. There are many publicly available disk-wiping tools of which Blancco 5, DBAN, and WipeDisk are well known. Recent research show that the tools are easy to use and subsequent data retrieval is quite hard (Lillis et al., 2016).
- Disk degaussing overwrites the data magnetically with zeros or random values. It is difficult today, but on older magnetic disks the Gutmann patterns enable investigators to retrieve data by using magnetic force microscopy (Garfinkel & Shelat, 2003).
- Physical destruction techniques include melting, shredding, and incarnating (Kissel, Scholl, Skolochenko, & Li, 2012).
- File wiping is similar to disk wiping but focused on files. Sdelete is the most common tool (Conlan et al., 2016).

- Generic data wiping tools differ from file wiping tools by erasing artifacts like cookies, temporary data, and Internet browsing history. A well-known generic data-wiping tool is CCleaner (Conlan et al., 2016).
- Metadata wiping tools: Metadata is the data about the data. Metadata of a file stores times, ownership, size, etc. An example tool for metadata wiping is Timestomp. It is a part of the Mmetasploit framework (Garfinkel, 2007). Metadata wiping requires advanced knowledge and a successful exploit at the target system.
- The Windows registry is a database storing operating system and application-specific settings for the Microsoft Windows operating system. Registry wiping tools remove unused, broken, or wrong registry entries (Conlan et al., 2016).
- Removable-disk wiping uses similar techniques to that of disk wiping (Lillis et al., 2016).

3. Trail Obfuscation

This method is known as “counterfeiting” (Jain & Chhabra, 2014). Trail obfuscation adds misdirection to digital evidence (Harris, 2006). Misdirection includes timestamp modification, file defragmentation, and manipulation of log files. Any inconsistencies on those suggest a trail-obfuscation activity.

4. Attacks against Digital Forensic Tools and Processes

Conlan et al. (2016) described attacks against forensic tools. Analysis is the most important phase of forensic investigation, and file integrity is very important for a proper analysis. An attacker by detecting either image creation or analysis of the logical partitions (of files or directories) can alter the integrity of the evidence.

Denial of service is another attack type against forensic tools. By depleting resources like the RAM and CPU required by the tools, an attacker can impede analysis (Jain & Chhabra, 2014). Anti-reverse engineering is another method against forensic tools.

One way is compression bombs. Current tools open compressed files like "zip" files during the analysis of file system. Compression bombs are compressed files that when extracted gets bigger than the tool can handle, perhaps with recursively contained files.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ANALYSIS OF ANTI-FORENSIC TOOLS

We followed four steps for analysis of an anti-forensic tool: installation, configuration, usage, and analysis of the forensic artifacts on the victim system. As discussed above there are many anti-forensic tools. In choosing tools, the main criteria are effectiveness in circumventing forensics, availability, ease of usage, cross-platform capability, and resistance to cyber-attacks. More than 300 tools fit the criteria (Conlan, et al., 2016). Thus, we narrowed the focus of the choices by adding novelty, community support and popularity among the cyber criminals to the criteria list. Table 1 gives a short summary and Appendix A gives the full tool list considered.

Table 1. Anti-forensic Tools

Technique	Sub Category	Specific item analyzed here
Data Hiding	1. File System Data Hiding	BMAP and NTFS file streams
	2. Memory Data Hiding	Explanation of the techniques
	3. Network-based Data Hiding	Stunnel
	4. Encryption	VeraCrypt Whole disk and file encryption, VeraCrypt Hidden OS and plausible deniability
	5. Steganography	Audio
		Text using Hydan tool
		Image/video using home-developed program for gray-scaled pictures
		Protocol
	6. Rootkits	In general
	1. Disk Wiping	DBAN
Artifact Wiping	2. Disk Degaussing & Destruction	In general
	3. File Wiping	Sdelete and BitKiller
	4. Generic Data Wiping	CCleaner
	5. Metadata Wiping	Timestomp

Technique	Sub Category	Specific item analyzed here
	6. Registry Wiping	In general
	7. Removable Disk Wiping	In general
Trail Obfuscation		Log cleaners with the Metasploit framework.
Attacks against Forensic Tools and Techniques		Packers (7-zip, PECompact, UPX)

A. DATA HIDING

1. File System Data Hiding Tools and Techniques

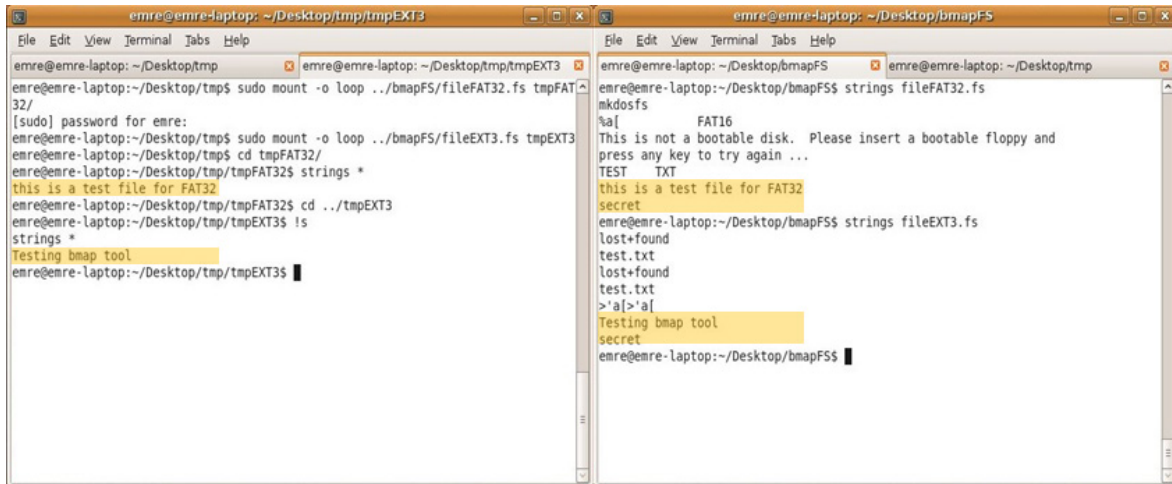
We chose the Bmap tool to investigate data hiding in slack space in UNIX environment. We installed Bmap version 1.0.17 on an Ubuntu 08.04 virtual machine with 1 core CPU, 1 GB memory, 10 GB HDD, and a bridged network. For testing, two image files were created using “dd” command and mounted on the Ubuntu file system. Then a string (“secret”) was put into the slack space. Appendix B explains Bmap usage and the commands that were used.

We extracted two image files from a virtual machine containing FAT32 and EXT3 file systems. The test images represented two computers with hidden data in their slack space. We hid a string into both file systems using the Bmap “putslack” option.

During the experiment, we altered to Bmap source code and recompiled it because it had not been updated since 2000. We compiled Bmap on Ubuntu 08.04 and created a text file using the command “echo ‘Testing bmap tool’ > text.txt.” The text.txt file had a large slack space at the end of the file-system block. We used “bmap --mode slack test.txt” to see the available slack space on the block. We ran “echo “secret” | bmap --mode putslack text.txt” to put the “secret” string into the slack space of the text.txt. Appendix B has a more detailed explanation.

Initial examination was done using LINUX “strings” command. Figure 2 shows that it can extract strings in the slack space of an EXT3 file system but not a FAT32 file system. Secondly, we tried forensic tools to extract the hidden strings from the slack space. The Autopsy tool, installed on a Windows machine-GUI, could not extract them for

both file system types, but FTK Imager, Bulk Extractor using BEViewer, and TSK could extract it.



The image shows two terminal windows side-by-side. The left window is titled 'emre@emre-laptop: ~/Desktop/tmp/tmpEXT3' and shows the following commands and outputs:
1. `sudo mount -o loop ../bmapFS/fileFAT32.fs tmpFAT32/` - outputs '[sudo] password for emre:'
2. `sudo mount -o loop ../bmapFS/fileEXT3.fs tmpEXT3`
3. `cd tmpFAT32/`
4. `strings *` - outputs 'this is a test file for FAT32'
5. `cd ../tmpEXT3`
6. `strings *` - outputs 'Testing bmap tool'
The right window is titled 'emre@emre-laptop: ~/Desktop/bmapFS' and shows the following commands and outputs:
1. `strings fileFAT32.fs` - outputs 'mkdosfs', 'FAT16', 'This is not a bootable disk. Please insert a bootable floppy and press any key to try again ...', 'TEST', 'TEXT', 'this is a test file for FAT32', 'secret'
2. `strings fileEXT3.fs` - outputs 'lost+found', 'test.txt', 'lost+found', 'test.txt', '>[a]>[a]', 'Testing bmap tool', 'secret'

Figure 2. FAT32 and EXT3 Strings Command Output

For data hiding in NTFS file systems, we used the NTFS alternate data streams tool (ADS). NTFS metadata includes the files as shown at the Table 2; \$DATA, \$Boot, \$BadClus and \$MFT provide opportunities to hide data (Huebner et al., 2006). ADS constructs and alters metadata of NTFS files. It cannot be detected with forensic tools because it contains title, author, and other necessary metadata about a file. Forensic tools usually do not check the metadata.

Table 2. Metadata Files in NTFS. Adapted from Huebner et al. (2006).

0	\$MFT	MFT 512B
1	\$MFTMirr	Backup MFT
2	\$Log	Transaction logs
3	\$Volume	Volume information
4	\$AttrDef	Attribute definition
5	.(dot)	Root directory of the system
6	\$BitMap	Allocation status of all clusters
7	\$Boot	Boot record
8	\$BadClus	List of bad clusters
9	\$Secure	Security and access control information
10	\$Upcase	Converts lowercase characters to Unicode uppercase
11	\$Extend	Extension directory
12-15	Unnamed	For future use
24	\$Extend	

The System Internals tool “lads.exe” can detect many data streams but not alternate ones. However, “streams.exe” and “streams64.exe” can detect them. Huebner et al. (2006) observed that detecting alternate data streams is laborious job and a forensics examiner can have difficulty discriminating between legitimate and malicious usage. A stream needs to be examined carefully with a hex editor. So alternate data streams are hard to conceal, and are more an obfuscation technique than a data-hiding technique.

For the experiment, we created a pointer “evil.exe” in \$DATA attribute of file normal.exe. This pointer can be detected using System Internals “streams64.exe” tool. Figure 3 shows that “streams64.exe” can extract the pointer in “normal.exe” which hides “evil.exe” although it resides in a different cluster of the NTFS. The pointer can also be in either the “author” or “title” attribute of the NTFS file so it will not attract attention during the forensic analysis.

```

C:\Users\emrec\Desktop\ADS>type D:\evil.exe > .\normal.exe:a
C:\Users\emrec\Desktop\ADS>..\Forensics\Streams\streams64.exe .\normal.exe

streams v1.60 - Reveal NTFS alternate streams.
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Users\emrec\Desktop\ADS\normal.exe:
      :a:$DATA 37888
      :evil.exe:$DATA 37888
C:\Users\emrec\Desktop\ADS>

```

Figure 3. Testing Alternate Data Streams

2. Network-Communications Data Hiding

Stunnel provides network-based data hiding of unencrypted traffic in an SSL-protocol traffic using a self-signed certificate. Stunnel needs a server and a client. The user creates an SSL certificate using the OpenSSL library, and both server and client use the certificate. Appendix B has details about the installation and configuration. For testing, the topology shown in Figure 4 was established. The Netcat message-exchange utility was used to transfer data.

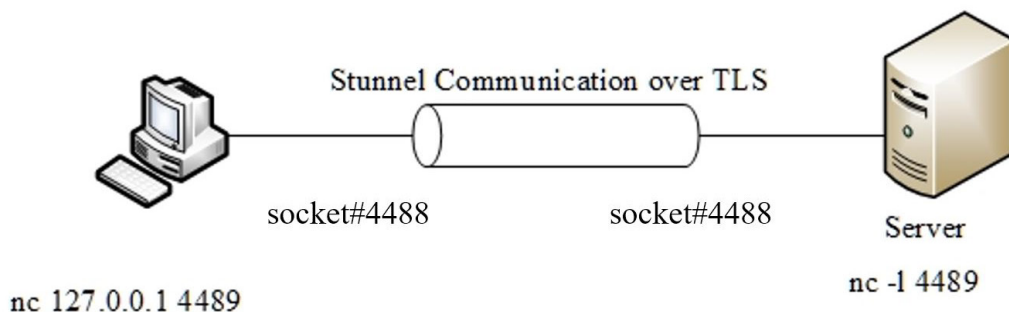


Figure 4. Stunnel Topology Used for Experiments

For the experiment, we generated symmetric encryption keys using OpenSSL. Keys were exchanged between the client and the server by another means of communication (IRC messaging) or manually. After that, we simulated insecure network communication using the “netcat” tool. The client started “netcat” on the localhost port 4489. Stunnel was

configured to get the messages on port 4489 and send them in SSL tunnel through the network interface. On the server side, a similar configuration was used; Netcat configured to listen on port 4489 and Stunnel was configured to get SSL messages and pass them to localhost port 4489. Appendix B has the configuration and terminal commands used for this experiment.

Initially we analyzed the Stunnel communication between client and server. We used Wireshark, an open-source tool which captures network packages and analyzes them. Our aim was to capture TLSv 1.2 communication establishment messages. Those messages showed a client-server communication with a predefined symmetric key, which is an uncommon use of TLS. Normally, TLS uses PKI for exchanging keys between the server and the client.

When the Stunnel communication ended, the only artifacts remaining were the installation packets and configuration files on the client and the server machine. TSK, FTK Imager, Magnet Forensics AXIOM, and Linux “grep” (with appropriate search strings) can detect and display installation and configuration files from the image files of the client and the server machines. However, if an attacker wants to better deceive a forensic investigator, they can recompile the Stunnel source under a different name.

A key indicator of Stunnel is its certificate. The Stunnel certificate includes both the private key and certificate file in a single PEM file, which begins with a “BEGIN RSA PRIVATE KEY” string. This string shows that private and public keys are concatenated back to back. On the other hand, the certificate-authority signed certificate starts with “BEGIN CERTIFICATE” and it is in PKCS format. Figure 5 shows Stunnel and Microsoft-signed certificates to show starting statements and contents.

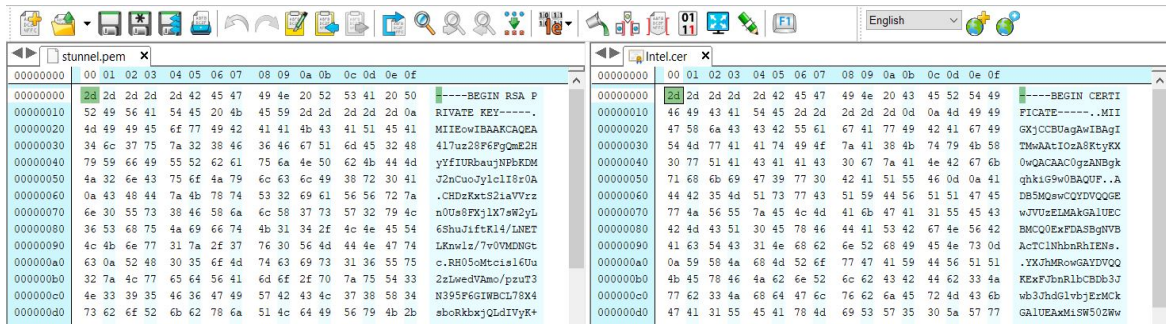


Figure 5. Stunnel Private Key and Certificate and Certificate-Authority Signed Certificate

Since Stunnel can also provide privacy to a legitimate communication, a forensic investigator needs to examine its configuration file. Figure 6 shows an example configuration file which secures HTTP traffic between a client browser and IP Address 172.217.169.142 (www.google.com) on the port 443. A forensic investigator can examine the Stunnel configuration file to see if it was used for malicious purposes.

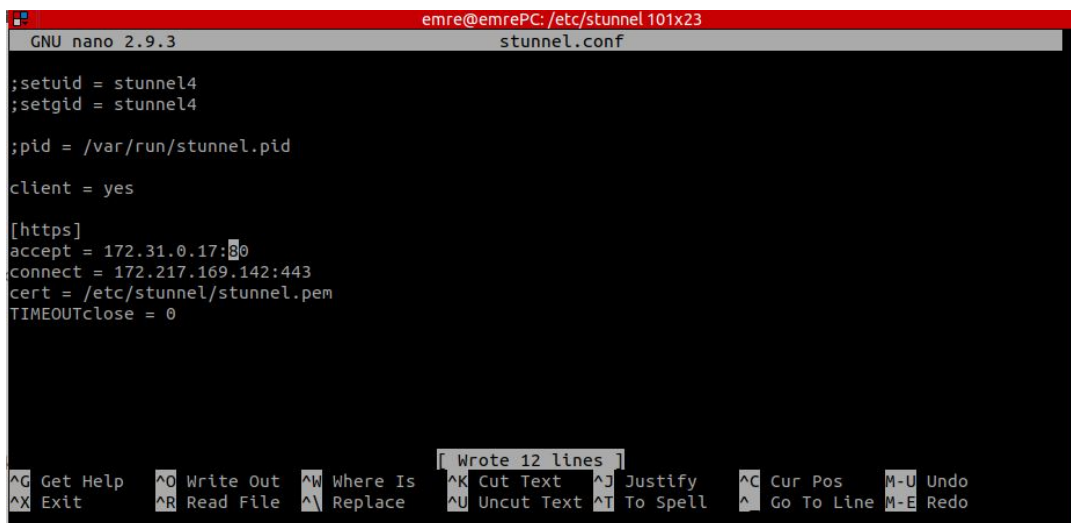


Figure 6. The Stunnel Configuration for Improving Security

3. Using Encryption for Data Hiding

A user can use encryption to hide data from a forensic investigation. We investigated the VeraCrypt encryption tool, which provides file, partition, and volume encryption (Idrix, n.d.). Our experiments used both standard and hidden volumes with 100 MB disk space. The standard volume of VeraCrypt is an encrypted file container, which mounts itself when correct password is input to VeraCrypt. The hidden volume is a standard volume which contains another standard volume, and mounts itself only when the password of the hidden volume is input. If the user inputs the standard volume's password, the hidden volume remains unmounted and standard volume mounts itself. This protective standard volume is called "outer" disk space/volume of the hidden volume. The hidden volume included 50 MB of "outer" disk space and the rest was 50 MB in size in our experiments

In experiments, we stored three text files in the standard and hidden volumes. Our experimental environment was Windows 10 OS running on virtual platform of Oracle VBox. To examine the VeraCrypt tool and volumes, a VMDK-file to binary-file conversion was done. Hexadecimal value analysis of the two volume files showed that neither contained successive zero bytes. Filling out empty parts of the volume is a feature of VeraCrypt (Idrix, n.d.). However, empty parts in a file must be filled with zero bytes in Windows 10; seeing no successive zero bytes is a clue to use of VeraCrypt. Figure 7 shows that TSK-Autopsy forensic analysis tool tags volume files as possible encrypted files.

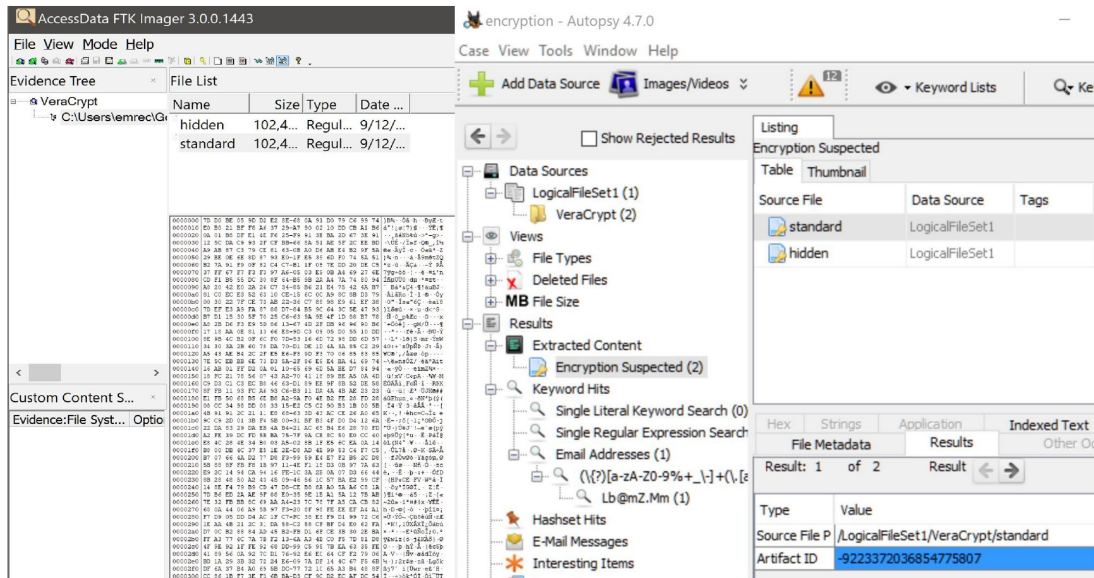


Figure 7. Forensic Analysis of VeraCrypt Volumes with TSK Autopsy and FTK Imager

Detection of encrypted data in the hidden volume is harder than in the standard volume. The hidden volume provides deniable encryption and plausible deniability (Kedziora, Chow, & Susilo, 2017). This meant that the first part (outer volume) must hide the second part (hidden volume), and an investigator cannot collect information about second part by accessing the first part (Kedziora et al., 2017). However, after mounting the outer volume, entropy analysis of the mounted volume showed where the hidden volume starts. Kedziora et al. (2017) suggested that dramatic drops on entropy values indicates start and end points of hidden volume because the header files of the VeraCrypt volumes do not contain random data when compared with the whole image, even when a hidden volume created in a standard volume. Therefore, bit-entropy analysis of a VeraCrypt file yields information about the existence of a hidden volume.

4. Steganography and Example Implementation

We developed our own steganography application. It hides an image in another one by a substitution technique, the most common technique for text hiding (Mishra et al., 2014). Substitution methods that are useful for text steganography include least-significant-bit, random, and specific-bit replacements. Least-significant-bit image

steganography technique hides data into least-significant bits of image data, and there are many options on which bytes to change. We developed an image steganography program, which hides an image into a base image after encrypting it. It can hide half a grayscale secret image in a base image so that the human eye cannot detect the difference. Appendix B shows our implementation of this algorithm. Figure 8 displays the user interface to the tool.

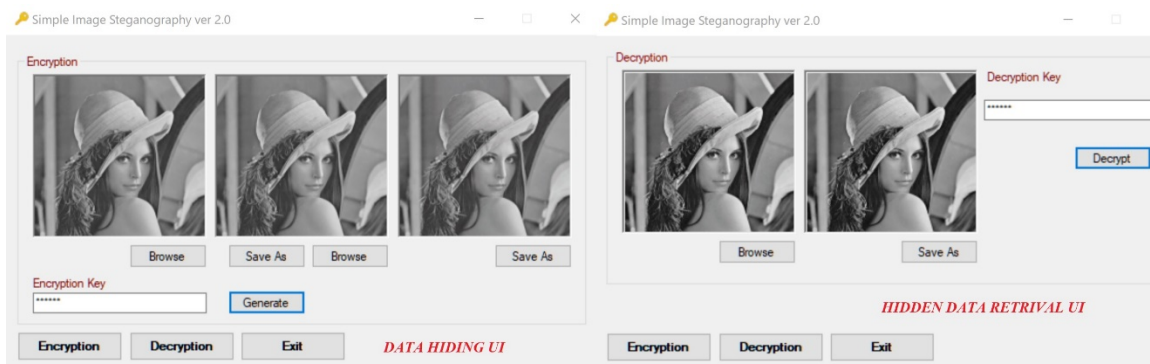


Figure 8. Data Hiding and Retrieval with a Simple Image Steganography Tool

When a forensic analyst suspects the existence of steganography in an image file, it is helpful to examine the hexadecimal values of the bytes of the image files. Statistical analysis can suggest the existence of steganography in a file. Shannon (2001) suggests that entropy (H) can help us to understand the uncertainty of the information source, so unusual values for the entropy of a file can indicate steganography. However, examining each file and trying to recognize a particular implementation of steganography is still challenging.

We tested steganography detection with two images in BMP format. We inserted a secret message into these files. However, hexadecimal analysis showed the resulting images were in PNG format (according to the Linux “file” command after examining their headers). AforgeNet.dll was used in the C# source code and it saves only in PNG format. Appendix B has the source code for the image steganography tool.

We also wrote a Python program which creates a histogram of Shannon entropy values for the bytes. The original image for the example picture shown above has a byte

distribution shown in Figure 9. The image with steganography has fewer distinct values and no values over 127 (Figure 10). The differences suggest use of steganography.

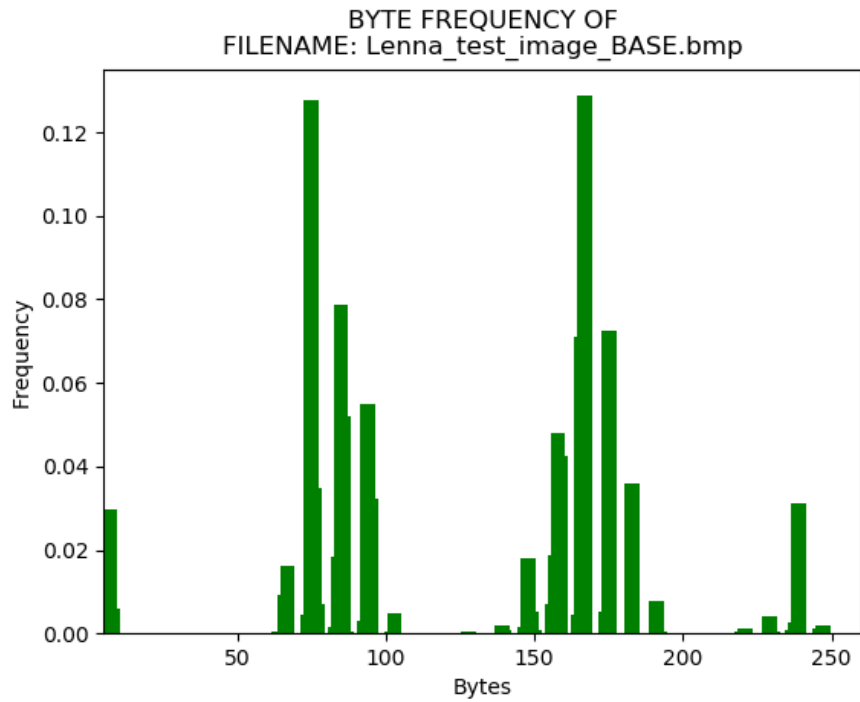


Figure 9. Histogram of Base Image in Test

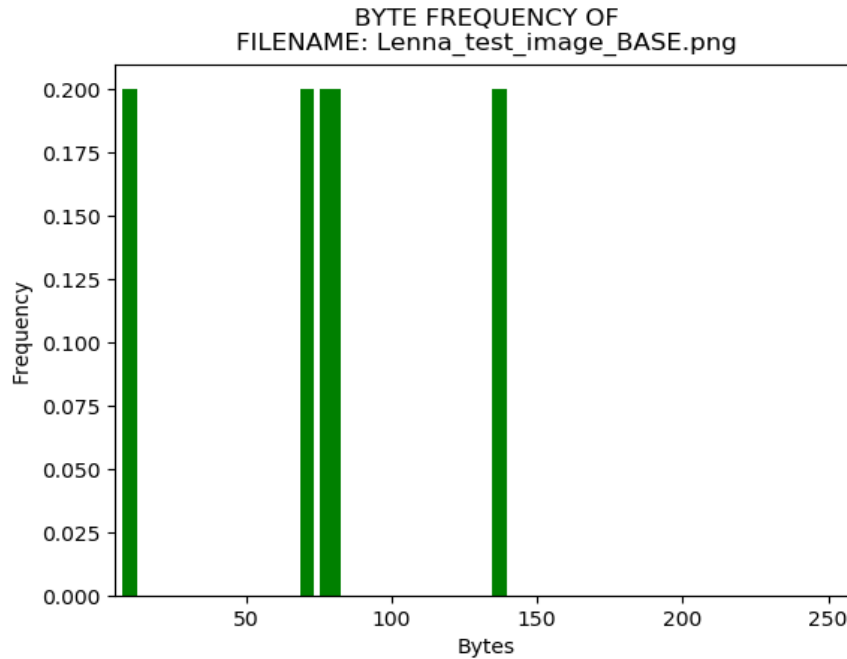


Figure 10. Histogram of Steganography Image in Test

B. ARTIFACT WIPING TECHNIQUES

1. File Wiping

We first tested a carving on a file marked as deleted. A file in text format was created on an Ubuntu 18.04 OS using FTKImager. Forensic analysis of the image found the file and all its metadata and data could be recovered as expected. After deleting the file, we used the “dd” tool to make a logical image of the drive by the command “sudo dd if=/dev/sda1 of=/media/sf_VM_share/shred1.raw bs=4096 conv=noerror,sync”. The Foremost, Scalpel, Autopsy, FTK, and FTKImager tools successfully retrieved the deleted file.

Next, we used the Shred tool on the text file. The tool was installed on the Ubuntu 18.04 using “sudo apt-get install shred” command. Then the Shred tool’s zeroing option was used once as seen in Figure 11. When contents of the file were retrieved using the “cat” command, nothing appears in the terminal window. Similarly, the forensic-tool analysis of the file showed that its size was zero and its contents were all zeros.

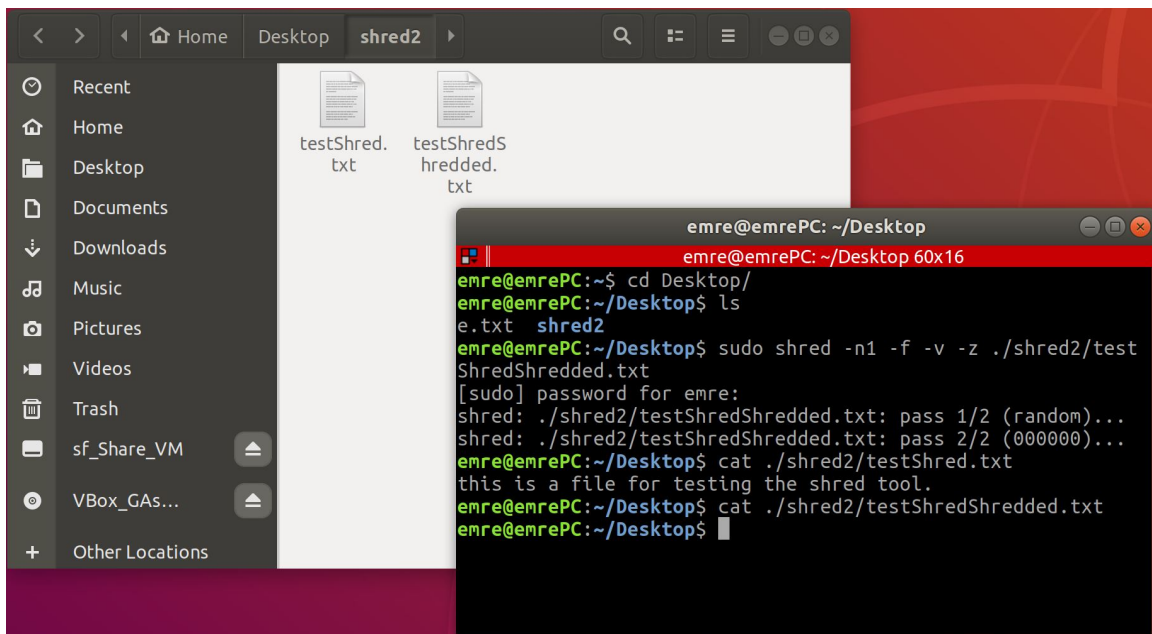


Figure 11. Output of the Shred Tool on a Wiped File

We also examined the hexadecimal values in which the shredded file formerly resided. Examination provided no clues about the presence of the file. However, logs and folder names do report to the forensic analyst that Shred was used on a system.

Sdelete and BitKiller deleted and overwrote files as well. Both tools provide DoD-secure deleting options, and both are freeware. For testing the programs, we downloaded them to a virtual machine, a Windows 10 build 1809. Test folders were created with a text file. One copy of the file was deleted with Sdelete using the overwrite option, one copy was deleted with BitKiller, and one copy was deleted using the Windows File Explorer. We then created a raw logical disk image of “C” drive with FTK Imager, and used TSK, Autopsy, AXIOM, FTK Imager, and Foremost tools for analysis of the Windows image. No forensic tools could retrieve the deleted files from the image. However, memory analysis provides details about BitKiller if a memory image is made while BitKiller is running. Therefore Sdelete is better than BitKiller at concealment.

2. Generic-Data Wiping and Registry Wiping

Not only disks and files contain artifacts, but also web browsers, applications, and third-party tools. Generic wiping tools delete browser temporary Internet files, cookies,

caches, registry, and so on. One of the most common generic-artifact wiping tools is Piriform Ccleaner, a commercial Windows OS tool. According to Cnet.com, it has been downloaded nearly 161 million times. CCleaner overwrites a file at least three times with random data using the “rand ()” function in Windows to make the file data random. However, CCleaner is easy to see in a disk image since it creates an “.INI” file for storing configuration data under the directory “C:\Program Files\CCleaner.” Another indicator is the prefetch data mentioning “ccleaner” and “piriform.”

3. Metadata Wiping

We tested metadata wiping using the Metasploit Framework. The Experimental details and setup are in Appendix B. In our experiment, an attacker deleted timestamps of the file to conceal his tracks on the exploited system. We examined the file timestamps, saw big inconsistencies between the usual OS files and the manipulated ones. This is an indicator for metadata wiping activity.

C. TRAIL OBFUSCATION

We did a trail obfuscation experiment on the same setup we used for metadata wiping. Trail obfuscation adds extra information or takes out information from a system to mislead a forensic examiner. In our experiment, we deleted all event logs from a Windows system. This is a major indicator for trail obfuscation activity on the system. But a cyber-criminal can also change some logs or create extra ones to further cover his tracks.

D. ATTACKS AGAINST FORENSIC TOOLS AND PROCESSES

Packers are file-compression tools and they can be used against forensic tools to provide code and data obfuscation. To analyze packers, we created a simple C code segment to print a sentence to the terminal window, then packed it with PECompact, 7-zip and UPX. Our results showed these packers compressed up to 70%. We ran the forensic tools FTKImager, Autopsy, TSK, AXIOM, and BelkaSoft. They all successfully detected that the compressed file contained an executable. When details of the executable were examined with IDA PRO, it was seen that the “magic number” identifying the type of file remained untouched. So packing alone is not a useful anti-forensics technique.

IV. MITIGATION OF ANTI-FORENSICS AND RECOMMENDATIONS

We analyzed four kinds of anti-forensics. Table 1 summarized our analysis methodology. Our analysis suggested that common anti-forensics techniques and tools can be detectable by their leaving important evidence material in various places. In this chapter tool-specific detection methods and mitigation techniques are presented.

A. DATA HIDING

1. Detection of File System Data Hiding

We installed and used the Bmap tool in a Linux environment. Detection of the tool can be done by the “strings” tool for the EXT3 file system. If the file-system format is FAT32, then detection can be done by the forensic tools FTKImager, Bulk-Extractor, and TSK.

A second file system data-hiding method is using ADS on an NTFS environment. Detection of ADS can be done using Microsoft System Internals “stream64.exe” tool. Our tests on ADS showed that it successfully detected a hidden stream in an executable that was pointing another malicious executable where the stream value was stored in \$DATA attribute of the executable. However, a forensic analyst can easily miss that pointer. A more novel method for detection of ADS is to use PowerShell with following steps:

- Collect the user-created files.
- Run “Get-item -Path [file_path] -Stream * | Export-Csv <.csv_path> “PowerShell cmdlet.
- Analyze the CSV files to detect uncommon ADS values.
- Run “Get-Content -Path <file_path> -Stream [uncommon_stream_name] >> Evidence_Streams.txt” PowerShell cmdlet. Common stream names such as \$DATA, \$AUTHOR, and \$FILE_NAME are stored in the metadata of the file.

Current state-of-art forensic techniques do not provide a mitigation technique for ADS, because it has many valid usages. However, our PowerShell detection method can be turned into a script, which runs on the client and sends stream contents to a server for further analysis.

2. Detection of Network Communication Data Hiding with Stunnel

We tested Stunnel for data hiding. Stunnel communication is quite secure because it uses the SSL protocol. Network-traffic analysis did not reveal its hidden data. Communication initialization messages and socket communications (IP and port-tuple communications) were analyzed using Wireshark for network-traffic analysis, but there were no major indicators of the data hiding using Stunnel. There were clues in Stunnel configuration files, server-client TLS communication for high-end (>1024) ports, use of the OpenSSL library certificate, string search revelations of Stunnel keywords like stunnel, stunnel4, stunnel.conf, etc.), and the “var/run/stunnel.pid” file.

Stunnel runs on the Linux. Stunnel requires OpenSSL library, a designated “uid” and “gid” pairs for Stunnel, and /var/run/stunnel.pid file, containing the “pid#” for Stunnel. A good mitigation against Stunnel is restricting users from creating or changing the Stunnel installation and configuration requirements.

3. Detection and Mitigation Techniques against Encryption Usage for Data Hiding

Encrypted files are hard to analyze without the key. Sometimes in a criminal investigation the key can be retrieved from the suspect. On the other hand, if analyst cannot detect any encrypted files, then they need to put mitigation techniques beforehand.

Our experiments on VeraCrypt revealed that TSK-Autopsy did flag VeraCrypt volumes as encrypted files. Mitigation techniques against this method for the enterprise-level networks are:

- A good corporate file-server policy so users cannot map a file share, and only GPO or scripts can.

- Effective device control metrics. When a user plugs in a removable media, contents should be copied to a central location (an evidence folder) for examination.
- Installing an executable must be disabled. If the user is a system administrator, installing the executable must be logged and monitored. It is important to prepare a master operating-system image containing all the required programs and executables. If a new executable is required, it must get approval from the change-management board.
- Users should access encryption libraries such as OpenSSL only with approval. and encryption must be done when required at the background and not by users.

4. Detection of Steganography

Detection of the steganography depends on statistical analysis of the image files. We discussed image steganography in Chapter III and suggested using Shannon entropy for detecting anomalies in the cover image. However, this does require having both the base image and the final image

B. ARTIFACT WIPING

Detection of artifact wiping is easy by observing data patterns of 0s or 1s; however, retrieving the deleted data is cumbersome, and usually not possible. Mitigation methods against artifact wiping are thus important Mitigation techniques against the artifact wiping are possible for enterprise networks where there is a central server that maintains and administers the network. The server can employ rules to control user activity. Some things they need to manage are:

- Artifact-wiping tools are cleaning tools that delete registry, temp files, browser history, and so forth. System administrators must prevent users from downloading and installing such tools. The list of tools at the Appendix A can serve as a guide.

- User-activity logs are valuable for detecting artifact-wiping activity. They can be saved in a SIEM log-collection database for integrated analysis.
- System administrators need to prevent users from accessing system root files and folders. This includes the Windows “C:” drive and in Linux all the directories except user’s home director.
- The ideal option is live forensic-artifact collection using an agent-based forensic application on the client systems.

C. MITIGATION TECHNIQUES AGAINST TRAIL OBFUSCATION AND ATTACK AGAINST FORENSIC TOOLS

Trail obfuscation tools misdirect the forensic analysis. Cyber criminals use it in the post-exploit phase of an attack. The best mitigation against the trail obfuscation is protecting the systems against a cyber-attack.

We tested zip bombs and packers as attack methods against the forensic tools FTK Imager, TSK-Autopsy, and Magnet Forensic AXIOM. They can detect zip bombs and recover themselves against this attack. Success of the packers depends on the technique used. If a packaging method like UPX is used against forensic tools, detection is possible because UPX cannot hide the contents of the executable. On the other hand, the 7-Zip tool encrypts both contents and file names, so this tool is effective against initial forensic analysis. However, forensic examiners can provide findings of encryption and legal authorities can request keys and passwords.

D. SUMMARY

Anti-forensics techniques and tools are ever-changing. Therefore, a solid set of forensic tools is required combatting them. We first examined anti-forensic techniques and identified tools in a number of categories. We identified publicly available, popular, up-to-date, and easy-to-use tools. Our experiments on selected tools showed that anti-forensic techniques do complicated well-known forensic practices. To detect each tool, forensic examiner must look different parts of the operating system and must follow different

methodologies. We suggested mitigation and detection techniques that can help forensic examiners to prevent anti-forensics or to detect its use.

Important threats we have not examined are rootkits and malware activity. Future work should investigate mitigation and detection methods for them following our methodology.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. ANTI-FORENSIC TOOLS CONSIDERED

Conlan et al. (2016) provided a list of 308 anti-forensic tools. A tool that supports multiple operating system and is an open-source, easy to download, easy to configure, and up-to-date was considered a good candidate for analysis.

<i>Tool Name</i>	<i>Short Description</i>
<i>AbsoluteTelnet</i>	Hiding tool. Network-based, Telnet emulator, Windows, up-to-date, commercial, http://www.celestialsoftware.net/ .
<i>Ace encrypt</i>	Encryption. Windows and Linux, up-to-date, both freeware and commercial, no site available.
<i>Acid Cryptofiler</i>	Encryption. Windows, outdated (2008), http://acid-cryptofiler.software.informer.com/7.1/ .
<i>AdvFS</i>	Encryption. File-system level, Open-sourced, up-to-date.
<i>AES Crypt*</i>	Encryption. Windows, Linux, MacOS, mobile platforms. Open-source, up-to-date, file-based encryption support, https://www.aescrypt.com/ .
<i>Aloaha Crypt Disk</i>	Encryption. Full-disk, Windows, up-to-date, http://www.aloaha.com/aloha-crypt-disk/ .
<i>Aloaha PKCS #7 Crypter</i>	Encryption. Windows, up-to-date, http://www.aloaha.com/aloha-security-solutions/aloha-crypt-sign-and-zip/aloha-pkcs-7-crypter/ .
<i>Aloaha USB Endpoint Security</i>	Several features.. Partially uses data hiding, partly implements data-loss-protection. http://www.aloaha.com/aloha-security-solutions/aloha-usb-monitor/ .
<i>Android Privacy Guard</i>	Encryption. Latest version is 2014, https://github.com/thialfihar/apg .
<i>AxCrypt</i>	Encryption. Windows, MAC, IOS, Android. free and commercial, up-to-date https://www.axcrypt.net/ .
<i>Bcrypt</i>	Encryption. Open-source, old, not practical http://bcrypt.sourceforge.net/ .
<i>BestCrypt Container Encryption</i>	Encryption. Commercial, up-to-date, Windows https://www.jetico.com/ .
<i>BestCrypt Volume Encryption</i>	Encryption. Commercial, up-to-date, Windows https://www.jetico.com/ .
<i>Bitlocker</i>	Encryption. Windows.
<i>Bmap*</i>	Slack space hiding tool. Linux, open-source, obsolete. https://packetstormsecurity.com/files/17642/bmap-1.0.17.tar.gz.html .

<i>Tool Name</i>	<i>Short Description</i>
<i>Boringssl</i>	Encryption. Data-in-transit, Forked Google developed SSL library.
<i>Botan</i>	Encryption. Cross-platform, up-to-date, https://botan.randombit.net/ .
<i>Bouncy Castle</i>	Encryption. C#, open-source, cross-platform https://www.bouncycastle.org/ .
<i>Burneye</i>	Encryption. https://github.com/packz/binary-encryption/blob/master/binary-encryption/burneye-stripped/src/burneye.c .
<i>Ccrypt</i>	Encryption. Files and streams. Outdated, Windows and Linux, http://ccrypt.sourceforge.net/ .
<i>Challenger*</i>	Encryption. Files, portable and local usage, secure deletion, Windows, up-to-date, freeware and commercial, http://www.encryption-software.de/challenger/en/index.html .
<i>CipherShed</i>	Encryption. Project forked from TrueCrypt, community failed to get support, cross platform, https://ciphershed.org/ .
<i>CloudFogger</i>	Encryption. Files, cloud storage services, background working, cross platform, freeware, up-to-date, https://cloudfogger.en.softonic.com/?ex=REG-60.2 .
<i>CrossCrypt*</i>	Encryption. CD/DVD encryption property is interesting, old, Windows, http://www.softpedia.com/get/Security/Encrypting/Cross-Crypt.shtml .
<i>Cryptainer</i>	Encryption. Windows, Up-to-date, freeware and commercial, https://www.cypherix.com/cryptainerle/ .
<i>Cryptarchiver</i>	Encryption. Folders and disks. Windows, up-to-date, commercial.
<i>Cryptmount*</i>	Encryption. Mounts encrypted file systems without privileged user accounts, Linux, up-to-date, library, http://cryptmount.sourceforge.net/ .
<i>CryptoCat*</i>	Encryption. Chat program, cross-platform, non-commercial, up-to-date, https://crypto.cat/ .
<i>Cypherix*</i>	Encryption. Splits hard drive into small containers and encrypts them, Windows, up-to-date, commercial, https://www.cypherix.com/cryptainerpe/ .
<i>DiskCryptor</i>	Encryption. Drives, Windows, freeware, https://diskcryptor.net/wiki/Main_Page .
<i>Dropbear</i>	Encryption. SSH-based server, data-on-transit encryption, Linux, up-to-date, https://matt.ucc.asn.au/dropbear/dropbear.html .
<i>eCryptfs</i>	Encryption. Drives and folders, Linux, up-to-date, open-source, http://ecryptfs.org/ .
<i>Encrypted File System on AIX</i>	Encryption. Files, Linux, up-to-date, open-source.
<i>Ergosecure</i>	Wiping and encryption. Cross-platform, up-to-date, commercial. https://ergosecure.com/en/ .
<i>EncFS</i>	Encryption. Files, open-source, Linux, up-to-date, https://github.com/vgough/encfs .
<i>EncryptStick</i>	Encryption. Cross-platform, commercial, up-to-date, https://www.encryptstick.com/ .

<i>Tool Name</i>	<i>Short Description</i>
<i>GPG</i>	Encryption. Open-source, Linux, up-to-date, https://gnupg.org/ , similar applications are Gpg4win and GPGmail.
<i>Hcovert*</i>	Steganography. Covert channel network data hiding tool, https://sourceforge.net/projects/hcovert/ .
<i>HICCUPS (Hidden communication system for corrupted networks)</i>	Steganography. Link layer, requires CSMA/CD or CA channel access to read all the sent messages to retrieve hidden secret, http://krzysiek.tele.pw.edu.pl/pdf/acs2003-hiccups.pdf .
<i>Hushmail</i>	Encryption and safe deletion. Cross platform, commercial, up-to-date, https://www.hushmail.com/ .
<i>Hydan</i>	Steganography. Hides data into program executables, it is old but still works, Linux and Windows, http://www.crazyboy.com/hydan/ .
<i>I.CX</i>	Encryption. Network, works on a web browser, https://i.cx/?icx.screen=home&convId=0 .
<i>IAI-JCE</i>	Encryption. Cross-platform, commercial, https://jce.iaik.tugraz.at/ .
<i>Imagespyer G2*</i>	Steganography. Windows, freeware, up-to-date, http://qpdownload.com/imagespyer-g2/ .
<i>Invisible Secrets</i>	Encryption and steganography. Windows, commercial, up-to-date, http://www.invisiblesecrets.com/ .
<i>Keepass</i>	Encryption. Passwords, Cross-platform, up-to-date, open-source, https://keepass.info/ .
<i>Lastpass</i>	Encryption. Passwords and files, Cross-platform, up-to-date, freeware and commercial, https://www.lastpass.com/ .
<i>Libressl</i>	Encryption. OpenBSD and cross-platform, up-to-date, forked from OpenSSL, https://www.libressl.org/ .
<i>Loop AES</i>	Encryption. Drives, open-source, Linux, up-to-date, https://sourceforge.net/projects/loop-aes/ .
<i>LUKS manager</i>	Encryption. Drives, Linux, up-to-date, open source, https://sourceforge.net/projects/luks-manager/ .
<i>MagicFS*</i>	Steganography with encryption. Data hiding in a ext2/3 file systems, Linux, up-to-date, open-source, http://magikfs.sourceforge.net/ .
<i>MailClick</i>	Encryption. Web-based, cross platform, commercial and freeware, https://www.mailclick.com .
<i>MatrixSSL</i>	Encryption. Cross-platform, up-to-date, small fingerprint, https://github.com/matrixssl/matrixssl .
<i>Mbed TLS</i>	Encryption. Cross-platform, written in C, up-to-date, https://tls.mbed.org/ .
<i>metFS</i>	Encryption. File system, UNIX and OpenBSD, http://www.enderunix.org/metfs/ .
<i>Mitto Password Manager</i>	Encryption. Passwords, cross-platform, up-to-date, freeware, http://download.cnet.com/windows/mitto-password-manager/3260-20_4-10097870-1.html .

<i>Tool Name</i>	<i>Short Description</i>
<i>Mobistego</i>	Steganography. Android, LSB-based, freeware and commercial, https://sourceforge.net/projects/mobistego/ .
<i>Mujahideen Secrets</i>	Encryption. Windows, old, freeware, https://www.schneier.com/blog/archives/2008/02/mujahideen_sec_1.html .
<i>NetPGP</i>	Encryption. OpenBSD, cross-platform, http://netpgp.com/ .
<i>One big cloud (CloudZ)</i>	Encryption. For cloud file-sharing applications like Google Drive, Dropbox, freeware and commercial, https://cloudz.io/ .
<i>OpenPuff</i>	Steganography. Open-source, cross-platform, up-to-date, http://www.embeddedsw.net/OpenPuff_Steganography_Home.html .
<i>OpenSSH</i>	Encryption. SSH traffic, OpenBSD, cross-platform, up-to-date, https://www.openssh.com/ .
<i>OpenSSL</i>	Encryption. For TLS and SSL, open-source, cross platform, up-to-date, https://www.openssl.org/ .
<i>OurSecret</i>	Steganography. Windows, http://steganography.findmysoft.com/ .
<i>Password Safe</i>	Password encryption. Freeware, Windows, https://www.pwsafe.org/ .
<i>Peerio</i>	Encryption. Chat and cloud, freeware and commercial, web based, https://www.peerio.com/ .
<i>PEFS (Private encrypted file system) on FreeBSD</i>	Encryption. File systems for UNIX/Linux, open-source, up-to-date, PEFS (Private encrypted file system) on FreeBSD.
<i>Private Disk</i>	Disk encryption. Windows, commercial, https://www.dekart.com/products/encryption/private_disk/ .
<i>Proxy Crypt</i>	File encryption. Windows, up-to-date, open source, https://sourceforge.net/projects/proxycrypt/ .
<i>Quick Crypto</i>	Steganography and encryption. Windows, commercial, up-to-date, http://quickcrypto.com/download.html .
<i>Red JPEG XT</i>	Steganography. Windows, https://totalcmd.net/plugring/redjpeg.html .
<i>Rohos mini drive*</i>	Encryption. USB drive files, Windows, up-to-date, freeware https://www.rohos.com/products/rohos-disk-encryption/rohos-mini-drive/ .
<i>Sbwave</i>	Encryption. Email, freeware, Windows, http://www.sbwave.com/enkryptor/home.html .
<i>Scramdisk</i>	Disk encryption. Linux, open-source, http://www.securiteam.com/tools/5VP011F0BY.html .
<i>SecureDoc</i>	Encryption. Holographic data hiding, watermarking, commercial, cross-platform, https://www.winmagic.com/ .
<i>SypproofVPN</i>	Encryption. Data-in-transit, commercial, cross-platform, http://spyproof.net/ .
<i>StegFS</i>	Steganography. Linux and FreeBSD, open-source, https://sourceforge.net/projects/stegfs/ .
<i>strongSwan</i>	Encryption. Linux, open-source, https://strongswan.org/ .
<i>Stunnel*</i>	Encryption. Data-in-transit, Linux, Windows, open-source, https://www.stunnel.org/ .

<i>Tool Name</i>	<i>Short Description</i>
<i>Symantec Endpoint Protection</i>	Drive encryption. Windows, commercial, https://www.symantec.com/products/endpoint-encryption .
<i>TrueCrypt</i>	Encryption. Cross-platform, open-source, obsolete, http://truecrypt.sourceforge.net/ .
<i>USBCrypt</i>	Encryption. Flash drives, commercial, Windows, http://www.usbcrypt.com/ .
<i>Vera Crypt*</i>	Encryption. Cross-platform, forked from True Crypt, up-to-date, https://www.veracrypt.fr/en/Home.html .
<i>ViPNet Office</i>	Encryption. VPN, data-in-transit, Windows, http://vipnet-office.download3000.com/ .
<i>Win PT</i>	Encryption. Windows, open-source, up-to-date, http://winpt.wald.intevation.org/ .
<i>WolfSSL</i>	Encryption. Internet of things, TLS support, light weight library for C programming language, up-to-date, Gnu License, https://www.wolfssl.com/ .
<i>Zfone</i>	Encryption. Data-in-transit, VoIP, open-source, http://zfoneproject.com/ .
<i>ZFS</i>	Encryption. Linux and UNIX, open-source, http://zfsonlinux.org/ .
<i>Artifact Wiping Tools</i>	
<i>ACleaner</i>	Browser, application, and Windows registry data cleaner, up-to-date, freeware, http://www.cleansoft.com/cleaner/privacy_registry_cleaner.htm .
<i>Active Cleaner</i>	Disk wiping, Windows, http://download.cnet.com/Active-Eraser/3000-2092_4-10199620.html .
<i>Advanced System care Free*</i>	PC Cleaner, freeware, Windows, up-to-date, https://www.iobit.com/en/advancedsystemcarefree.php .
<i>Aevita Erase Hard Drive</i>	Hard disk cleaner, wiping tool, Windows, beta version, http://aevita-erase-hard-drive.en.lo4d.com/ .
<i>Aevita Wipe & Delete</i>	File and folder deletion tool, Windows, http://aevita-wipe-and-delete.en.lo4d.com/ .
<i>Aomei Partition Assistant</i>	Disk wiping, Windows, commercial, https://www.disk-partition.com/free-partition-manager.html .
<i>Argente Utilities*</i>	Registry fixing, disk wiping, shredding, Windows, freeware, https://argenteutilities.com/ .
<i>Ashampoo WinOptimizer</i>	Registry fixing, disk wiping, browser data cleaning, Windows, up-to-date, freeware, https://www.ashampoo.com/en/usd/pin/3606/system-software/winoptimizer-free .
<i>Auslogics Registry Cleaner</i>	Registry wiping, Windows, freeware, up-to-date, https://www.auslogics.com/en/software/registry-cleaner/ .

<i>Tool Name</i>	<i>Short Description</i>
<i>Baidu PC Faster</i>	Generic wiping tool, Windows, up-to-date, http://www.pcfaster.com/en/ .
<i>BCWipe</i>	Data wiping, cross-platform, up-to-date, commercial, https://www.jetico.com/downloads/data-wiping .
<i>BitKiller</i>	Data wiping, Windows, freeware, not very new, relatively basic tool, http://www.snapfiles.com/get/bitkiller.html .
<i>Blancco Tools</i>	Comprehensive data wiping tools for cross platform, commercial, up-to-date, https://www.blancco.com/ .
<i>CBL Data Shredder</i>	Data wiping, Windows, up-to-date, freeware, http://www.cbldatarecovery.com/data-shredder/ .
<i>CCleaner*</i>	Generic data cleaner, wiping tool, freeware and commercial, suitable for home-use, up-to-date, https://www.ccleaner.com/ccleaner .
<i>Cleanersoft Registry Fix</i>	Registry wiping, Windows, freeware, up-to-date, http://www.cleanersoft.com/registry_fix/free_registry_fix.htm .
<i>CyberScrub</i>	Firm provides wiping tool for generic data wiping, Windows, commercial, http://www.cyberscrub.com/ .
<i>DBAN*</i>	Hard disk wiping tool, cross-platform, freeware, up-to-date, https://dban.org/ .
<i>Dclasyf</i>	Disk wiping, Windows, command-line tool, not new, http://www.dmares.com/maresware/html/declasfy.htm .
<i>Delete Files Permanently</i>	Small size file deletion tool, Windows, commercial, http://download.cnet.com/Delete-Files-Permanently/3000-2248_4-10790111.html .
<i>DP Secure Wiper</i>	File wiping, Windows, freeware, up-to-date, https://www.ghacks.net/2008/05/09/dp-secure-wiper-removes-files-securely-from-your-system/ .
<i>East-tec Eraser</i>	Generic data wiper, Windows, commercial, up-to-date, https://www.east-tec.com/eraser/ .
<i>Eraser</i>	File and disk wiping, Windows, up-to-date, commercial, https://eraser.heidi.ie/ .
<i>Eusing Registry Cleaner</i>	Registry wiping, fixing tool, Windows, freeware, not new, http://www.eusing.com/free_registry_cleaner/registry_cleaner.htm .
<i>FCleaner</i>	Generic data wiping, Windows, freeware, not new, http://www.fcleaner.com/ .
<i>Free Easis Data Eraser</i>	Data wiping tool, Windows, freeware, obsolete, http://download.cnet.com/EASIS-Data-Eraser/3000-2092_4-75452799.html .
<i>Free Window Registry Repair</i>	Registry wiping, Windows, freeware, not new, http://download.cnet.com/Free-Window-Registry-Repair/3000-2086_4-10606555.html .
<i>Freeraser</i>	File wiping, Windows, freeware, not new, http://www.freeraser.com/

<i>Tool Name</i>	<i>Short Description</i>
<i>Glary Utilities</i>	Generic data wiping, registry cleaning tools, windows, freeware, up-to-date, https://www.glarysoft.com/ .
<i>Hard Disk Scrubber</i>	Disk cleaning tool, Windows, freeware, not new, http://summitcn.com/hdscrub.html .
<i>Hard Drive Eraser</i>	Complete hard disk wiping, Windows, freeware, http://www.harddriveeraser.org/ .
<i>Hardwipe</i>	Disk wiping, Windows, freeware, up-to-date, http://hardwipe.com/ .
<i>HDDerase</i>	Bootable disk wiping tool, cross-platform, not new, https://www.lifewire.com/hdderase-review-2619137 .
<i>JetClean</i>	Generic clean tool, Windows, freeware, not new, http://www.majorgeeks.com/files/details/jetclean.html .
<i>Macrorit Data Wiper</i>	Disk wiper, Windows, freeware, up-to-date, https://macrorit.com/free-data-wiper.html .
<i>Mini Tool Drive Wipe</i>	Drive wiping, Windows, up-to-date. freeware, https://www.minitool.com/free-tools/minitool-drivewipe.html .
<i>Ontrack Eraser Degausser</i>	Disk Destruction, Degaussing, https://www.ontrack.com/products/data-erasure/degausser/ .
<i>Pointstone Registry Cleaner</i>	Registry wiping, fixing, commercial, up-to-date, https://www.pointstone.com/download/ .
<i>Powertools Lite</i>	Generic wiping, disk wiping, registry fixing, up-to-date, Windows, commercial, https://www.macecraft.com/download/ .
<i>PrivaZer</i>	Cleaner and wiping tool, Windows, up-to-date, freeware, https://privazer.com/ .
<i>Protect Star Data Shredder</i>	Data wiping, Windows, commercial, up-to-date, https://www.protectstar.com/en/products/data-shredder .
<i>Registry Life</i>	Registry fixing, Windows, up-to-date, freeware, https://www.chemtable.com/RegistryLife.htm .
<i>Registry Recycler</i>	Registry wiping, freeware, Windows, up-to-date, https://www.registryrecycler.com/ .
<i>Registry Repair</i>	Registry fixing, Windows, freeware, up-to-date, http://download.cnet.com/Free-Windows-Registry-Repair/3000-2086_4-10606555.html .
<i>RegSeeker</i>	Generic erasing, registry fixing, tuning, Windows, up-to-date, http://www.hoverdesk.net/index.php .
<i>Remo Drive Wipe</i>	Drive wiping, Windows, freeware and commercial, up-to-date, https://www.remsoftware.com/remo-drive-wipe .
<i>Remo File Eraser</i>	File deletion tool, Windows, freeware and commercial, up-to-date, https://www.remsoftware.com/remo-file-eraser .
<i>R-Wipe & Clean</i>	Wiping tool for files, Windows, commercial, up-to-date, http://www.r-wipe.com/ .
<i>Sdelete</i>	System Internals tool from Windows, Wiping, Windows, freeware, up-to-date, https://docs.microsoft.com/en-us/sysinternals/downloads/sdelete .

<i>Tool Name</i>	<i>Short Description</i>
<i>Secure Clean</i>	Wiping tool, Windows, commercial, up-to-date, https://www.whitecanyon.com/home-products/secureclean .
<i>Secure Eraser</i>	Wiping, Windows, freeware and commercial, up-to-date, http://www.secure-eraser.com/ .
<i>Securely File Shredder</i>	File eraser, Windows, freeware, up-to-date, http://www.securely.co/ .
<i>SlimCleaner Free</i>	Wiping and generic file deletion, Windows, up-to-date, freeware, http://download.cnet.com/SlimCleaner-Free/3000-18512_4-75279939.html .
<i>System Mechanic Free</i>	Generic wiping, Windows, up-to-date, freeware and commercial, http://www.iolo.com/downloads/download-system-mechanic/ .
<i>Timestomp*</i>	Metadata wiping, Windows and Linux, open-source, https://sourceforge.net/projects/timestomp-gui/ .
<i>TweakNow RegCleaner</i>	Registry cleaner, Windows, freeware, up-to-date, http://www.tweaknow.com/RegCleaner.php .
<i>TweakNow SecureDelete</i>	Disk wiping tool. Windows, freeware, up-to-date, http://www.tweaknow.com/SecureDelete.php .
<i>Wise Care 365</i>	Generic wiping tool, Windows, freeware and commercial. Up-to-date, http://www.wisecleaner.com/wise-care-365.html .
<i>XT File Shredder Lizard</i>	File removal tool. Windows, freeware, up-to-date, http://www.lizard-labs.com/xt_file_shredder_lizard.aspx .
<i>Ya- wipe</i>	Disk degaussing, cross-platform, obsolete, http://freshmeat.sourceforge.net/projects/ya-wipe .
<i>Trail Obfuscation Tools</i>	
<i>Attention-deficit disorder (ADD)</i>	Memory obfuscation, https://code.google.com/archive/p/attention-deficit-disorder/ .
<i>Bitblinder</i>	Anonymous P2P, https://bitblinder.en.uptodown.com/windows .
<i>Fake Location</i>	GPS data obfuscation for mobile, http://download.cnet.com/Fake-Location/3000-12941_4-75463190.html .
<i>Filetopia</i>	P2P sharing, cross-platform, http://www.filetopia.org/ .
<i>GNUnet</i>	Secure P2P, https://gnunet.org/ .
<i>I2P*</i>	Anonymous networking, cross-platform, https://geti2p.net/en/ .
<i>I2P-bote</i>	Anonymous email service using I2P network, https://github.com/i2p/i2p.i2p-bote .
<i>IMule</i>	Anonymous P2P, https://imule.en.softonic.com/ .
<i>JonDonym</i>	Anonymous proxy and web surfing, https://anonymous-proxy-servers.net/en/index2.html .
<i>Marabunta</i>	Anonymous free-net P2P, http://marabunta.laotracara.com/english.php .
<i>MUTE</i>	Anonymous P2P network, file sharing, https://sourceforge.net/projects/mute-net/ .

<i>Tool Name</i>	<i>Short Description</i>
<i>Netsukuku</i>	Anonymous P2P, free-net project, https://github.com/Netsukuku/netsukuku .
<i>OFFSystem</i>	Highly connected P2P network, http://offsystem.sourceforge.net/ .
<i>OneSwarm</i>	Private P2P sharing, http://www.oneswarm.org/ .
<i>Perfect Dark</i>	P2P file sharing, http://kasumi.moe/pd/ .
<i>Retroshare</i>	Encrypted chat, mail, share using Tor / I2P, http://retroshare.net/ .
<i>StealthNet</i>	Anonymous file sharing, http://www.stealthnet.de/en_index.php .
<i>Stego Share</i>	File sharing using steganography, http://stegoshare.sourceforge.net/ .
<i>Tribler</i>	Anonymous, Tor-based P2P, https://www.tribler.org/ .
<i>Attacks against forensic tools and methods</i>	
<i>7-zip</i>	Program packer, hard to analyze, initially archive needs unpacking, http://www.7-zip.org/ .
<i>BitCrypter</i>	Program Packet, commercial, https://www.crypter.com/download.html .
<i>PECompact*</i>	Program packer, https://bitsum.com/portfolio/PECompact/ .
<i>UPX (Ultimate Packer for eXecutables)</i>	Executable packer, cross-platform, open-source, up-to-date, https://github.com/upx/upx/releases/tag/v3.94 .

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INSTRUCTIONS FOR ANTI-FORENSIC TOOLS

A. BMAP TOOL INSTALLATION, CONFIGURATION PROCESS AND USAGE EXAMPLE

BMAP (Robertson, 2003) can be downloaded from the packetstormsecurity website (the link is provided at the Appendix-A). After downloading the TARBALL, the following steps (listed below) are enough for correct compilation of the tool.

- `tar -vxzf bmap-1.0.17.tar.gz`
- `cd bmap-1.0.7`
- `make`
- `sudo ln -s /home/<user>/Desktop/bmap-1.0.17/bmap /sbin/bmap`
- `bmap --help`

If the last command displays options of bmap tool, then the installation is completed successfully. Using `dd if=/dev/zero of=fileEXT3.fs bs=1024 count=10240` creates a 10MB raw image file. Using `mkfs.ext3` changes the raw image file to ext3 file system type. Following that, a user can mount the ext3 file system to a temp directory, navigate to the mounted temp directory, and use the following bmap options to hide a “*secret*” string in the slack space.

- `echo “Testing bmap tool” > text.txt`
- `bmap --mode slack test.txt` (displays slack space)
- `echo “secret” | bmap --mode putslack test.txt`
- `bmap --mode slack test.txt` (displays same slack space in step-2)

On Windows one can repeat the same four steps and unmount file systems.

B. STUNNEL TOOL INSTALLATION, CONFIGURATION AND USAGE

For installing Stunnel, the user needs to download the corresponding package and follow the installation directives. The best practice for Stunnel is configuring a Linux host as the server and either Windows or Linux host as the client. Stunnel needs a Linux package management environment such as dpkg. The installation is:

- `apt-get update`
- `apt-get upgrade`
- `apt-get install stunnel4`

This provides just the application. Specific configurations, SSL certificate setup, and service adjustments are also required. The list below shows the additional steps.

- `vi /etc/stunnel/stunnel.conf`
- Change “ENABLED = 1” for auto-start.
- Generate a key using OpenSSL:
- `# openssl genrsa -out key.pem 2048`
- Create a certificate for SSL communication:
- `# openssl -req -new -x509 -key key.pem -out cert.pem -days 1095`
- `# cat key.pem cert.pem >> /etc/stunnel/stunnel.pem`

Customize the configuration according to the topology at hand. The Stunnel default configuration file has options for IMAPS, https, POP3s, and TLS communication. An example configuration for the server is:

- `setuid = stunnel4`
- `setgid = stunnel4`

- pid = /var/run/stunnel.pid (stunnel.pid needs to be created beforehand including pid#)
- client = no
- [https] (accurate service name is required)
- accept = 4488
- connect = 127.0.0.1:4489
- cert = / etc/stunnel/stunnel.pem
- An example configuration for Linux client is:
- setuid = stunnel4
- setgid = stunnel4
- pid = /var/run/stunnel.pid (stunnel.pid needs to be created beforehand including pid#)
- client = yes
- accept = 4488
- connect = 4489
- cert = / etc/stunnel/stunnel.pem (certificate needs to be transferred from server to client)
- /etc/init.d/stunnel4 restart (on both OSes).

C. IMAGE STEGANOGRAPHY TOOL SOURCE CODE

Program.cs /*main C# code */

using System;
using System.Windows.Forms;

```

namespace Image_Stego
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

Form1.cs /*Desktop form application C-sharp side*/

```

using System;
using System.Collections;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using AForge.Imaging.Filters;

namespace Assignment_2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void buttonBrowseSimple_Click(object sender, EventArgs e)
            {
                OpenFileDialog fileDiag = new OpenFileDialog();
                fileDiag.Filter = "Bitmap Image (.bmp)|*.bmp| Gif Image (.gif)|*.gif| JPG  
Image (.jpg) |*.jpg| Png Image (.png)|*.png";

                if (fileDiag.ShowDialog() == DialogResult.OK)
                {
                    pictureBoxSimple.ImageLocation = fileDiag.FileName;
                    buttonBrowseSecret.Enabled = true;
                }
            }
        }
    }
}

```

```

private void buttonBrowseSecret_Click(object sender, EventArgs e)
{
    OpenFileDialog fileDiag = new OpenFileDialog();
    fileDiag.Filter = "Bitmap Image (.bmp)|*.bmp| Gif Image (.gif)|*.gif| JPG
Image (.jpg)|*.jpg| Png Image (.png)|*.png";

    if (fileDiag.ShowDialog() == DialogResult.OK)
    {
        Bitmap image = new Bitmap(fileDiag.FileName);
        pictureBoxSecret.Image = ToGreyScale(image);
        buttonSaveAsGrey.Enabled = true;
    }
}

private void buttonExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

// converts RGB values to grey scale
private Bitmap ToGreyScale(Bitmap bitmap)
{
    int grey, i, j;
    Color color;
    for (i = 0; i < bitmap.Width; i++)
    {
        for (j = 0; j < bitmap.Height; j++)
        {
            color = bitmap.GetPixel(i, j);
            grey = (int)((color.R + color.G + color.B) / 3);
            bitmap.SetPixel(i, j, Color.FromArgb(grey, grey, grey));
        }
    }
    return bitmap;
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    if (textBoxKey.Text.Trim().Length < 4)
    {
        buttonGenerate.Enabled = false;
        errorProvider.SetError(textBoxKey, "Key length must be greater than
3.");
    }
    return;
}

```

```

else
{
    buttonGenerate.Enabled = true;
    errorProvider.SetError(textBoxKey, "");
}

try
{
    int.Parse(textBoxKey.Text);
    errorProvider.SetError(textBoxKey, "");
}
catch (FormatException)
{
    errorProvider.SetError(textBoxKey, "Key must be number.");
    return;
}
}
// Saves only in BMP format
private void buttonSaveAs_Click(object sender, EventArgs e)
{
    SaveFileDialog fileDiagSave = new SaveFileDialog();
    fileDiagSave.Filter = "Bitmap Image (.bmp)|*.bmp";

    if (fileDiagSave.ShowDialog() == DialogResult.OK)
    {
        pictureBoxResult.Image.Save(fileDiagSave.FileName);
    }
}

private void buttonDecryption_Click(object sender, EventArgs e)
{
    groupBoxEncryption.Visible = false;
    groupBoxDecryption.Visible = true;
}

private void buttonEncryption_Click(object sender, EventArgs e)
{
    groupBoxEncryption.Visible = true;
    groupBoxDecryption.Visible = false;
}

private void Form1_Load(object sender, EventArgs e)
{
    groupBoxDecryption.Visible = false;
    groupBoxEncryption.Visible = true;
}

```

```

    }

    private void BrowseDecrypt_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.Filter = "Bitmap Image (.bmp)|*.bmp|Gif Image (.gif)|*.gif|JPEG  
Image (.jpeg)|*.jpeg|Png Image (.png)|*.png ";
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            pictureBoxEncryptedImage.ImageLocation = ofd.FileName;
        }
    }
    // Creates a Byte Array
    private byte getByte(byte[] bits)
    {
        String bitString = "";
        for (int i = 0; i < 8; i++)
            bitString += bits[i];
        byte newpix = Convert.ToByte(bitString, 2);
        int dePix = (int)newpix ^ key;
        return (byte)dePix;
    }

    private byte[] getBits(byte simplepixel)
    {
        int pixel = 0;
        pixel = (int)simplepixel ^ key;
        BitArray bits = new BitArray(new byte[] { (byte)pixel });
        bool[] boolarray = new bool[bits.Count];
        bits.CopyTo(boolarray, 0);
        byte[] bitsArray = boolarray.Select(bit => (byte)(bit ? 1 : 0)).ToArray();
        Array.Reverse(bitsArray);
        return bitsArray;
    }

    int key = 0;
    private void ButtonGenerate_Click(object sender, EventArgs e)
    {
        Bitmap simple = new Bitmap(pictureBoxSimple.Image);
        Bitmap secretGreyScale = new Bitmap(pictureBoxSecret.Image);

        if (secretGreyScale.Height != simple.Height || secretGreyScale.Width !=  
simple.Width)
        {

```

```

        ResizeBilinear  resizeFilter  =  new  ResizeBilinear(simple.Width,
simple.Height);
        secretGreyScale = resizeFilter.Apply(secretGreyScale);
    }
    // Initialize
    Color pixelContainerImage = new Color();
    Color pixelMsgImage = new Color();
    // get key in Integer
    key = int.Parse(textBoxKey.Text);

    byte[] MsgBits;
    byte[] AlphaBits;
    byte[] RedBits;
    byte[] GreenBits;
    byte[] BlueBits;

    byte newAlpha = 0;
    byte newRed = 0;
    byte newGreen = 0;
    byte newBlue = 0;

    /* Image Encryption */
    #region Encryption

    for (int i = 0; i < simple.Height; i++)
    {
        for (int j = 0; j < simple.Width; j++)
        {
            pixelMsgImage = secretGreyScale.GetPixel(j, i);
            MsgBits = getBits((byte)pixelMsgImage.R);
            pixelContainerImage = simple.GetPixel(j, i);
            AlphaBits = getBits((byte)pixelContainerImage.A);
            RedBits = getBits((byte)pixelContainerImage.R);
            GreenBits = getBits((byte)pixelContainerImage.G);
            BlueBits = getBits((byte)pixelContainerImage.B);

            AlphaBits[6] = MsgBits[0]; AlphaBits[7] = MsgBits[1];
            RedBits[6] = MsgBits[2]; RedBits[7] = MsgBits[3];
            GreenBits[6] = MsgBits[4]; GreenBits[7] = MsgBits[5];
            BlueBits[6] = MsgBits[6]; BlueBits[7] = MsgBits[7];

            newAlpha = getByte(AlphaBits);
            newRed = getByte(RedBits);
            newGreen = getByte(GreenBits);
            newBlue = getByte(BlueBits);

```

```

        pixelContainerImage = Color.FromArgb(newAlpha, newRed,
newGreen, newBlue);
        simple.SetPixel(j, i, pixelContainerImage);
    }
}
pictureBoxResult.Image = simple;
buttonSaveAs.Enabled = true;
#endregion
}

```

```

private void btnDecrypt_Click(object sender, EventArgs e)
{
    Bitmap EncryptedImage = (Bitmap)pictureBoxEncryptedImage.Image;
    Bitmap hiddenImage = new Bitmap (EncryptedImage.Width,
EncryptedImage.Height);
    Color pixelToDecrypt = new Color();
    try
    {
        key = int.Parse(textBoxDecryptionKey.Text);
    }
    catch (FormatException )
    {
        MessageBox.Show("Key must be number. ");
        return;
    }
}

```

```

byte[] BitsToDecrypt = new byte[8];
byte[] AlphaBits;
byte[] RedBits;
byte[] GreenBits;
byte[] BlueBits;
byte newGrey = 0;
/* Image Decryption */
#region Decryption

for (int i = 0; i < EncryptedImage.Height; i++)
{
    for (int j = 0; j < EncryptedImage.Width; j++)
    {
        pixelToDecrypt = EncryptedImage.GetPixel(j, i);

        AlphaBits = getBits((byte)pixelToDecrypt.A);
        RedBits = getBits((byte)pixelToDecrypt.R);
        GreenBits = getBits((byte)pixelToDecrypt.G);
    }
}

```

```

        BlueBits = getBits((byte)pixelToDecrypt.B);

        BitsToDecrypt[0] = AlphaBits[6];
        BitsToDecrypt[1] = AlphaBits[7];
        BitsToDecrypt[2] = RedBits[6];
        BitsToDecrypt[3] = RedBits[7];
        BitsToDecrypt[4] = GreenBits[6];
        BitsToDecrypt[5] = GreenBits[7];
        BitsToDecrypt[6] = BlueBits[6];
        BitsToDecrypt[7] = BlueBits[7];

        newGrey = getByte(BitsToDecrypt);
        pixelToDecrypt = Color.FromArgb(newGrey, newGrey, newGrey);
        hiddenImage.SetPixel(j, i, pixelToDecrypt);
    }
}
pictureBoxExtractedImage.Image = hiddenImage;
buttonSaveAsFinal.Enabled = true;
#endregion
}

private void buttonSaveAsFinal_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "Bitmap Image (.bmp)|*.bmp|Gif Image (.gif)|*.gif |JPEG Image (.jpeg)|*.jpeg |Png Image (.png)|*.png ";

    if (sfd.ShowDialog() == DialogResult.OK)
    {
        pictureBoxExtractedImage.Image.Save(sfd.FileName);
    }
}

private void button3_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();
    sfd.Filter = "Bitmap Image (.bmp)|*.bmp|Gif Image (.gif)|*.gif |JPEG Image (.jpeg)|*.jpeg |Png Image (.png)|*.png ";

    if (sfd.ShowDialog() == DialogResult.OK)
    {
        pictureBoxSecret.Image.Save(sfd.FileName);
    }
}

```



```

f = open(sys.argv[1], "r")
byteArr = map(ord,f.read())
f.close()
fileSize = len(byteArr)
print('File size in bytes:')
print('fileSize')

# calculate the frequency of each byte value in the file
freqList = []
for b in range(256):
    ctr = 0
    for byte in byteArr:
        if byte == b:
            ctr += 1
    freqList.append(float(ctr) / fileSize)
# Shannon entropy
ent = 0.0
for freq in freqList:
    if freq > 0:
        ent = ent + freq * math.log(freq, 2)
ent = -ent
print('Shannon entropy (min bits per byte-character):')
print(ent)
print('Min possible file size assuming max theoretical compression efficiency:')
print (ent * fileSize), 'in bits'
print (ent * fileSize) / 8, 'in bytes'

import numpy as np
import matplotlib.pyplot as plt

N = len(freqList)
ind = np.arange(N) # the x locations for the groups
width = 1.00 # the width of the bars

#fig = plt.figure()
fig = plt.figure(figsize=(11, 5), dpi=100)
ax = fig.add_subplot(111)
rects1 = ax.bar(ind, freqList, width)
ax.set_autoscalex_on(False)
ax.set_xlim([0, 255])

ax.set_ylabel('Frequency')
ax.set_xlabel('Byte')
ax.set_title('Frequency of Bytes 0 to 255\nFILENAME: ' + sys.argv[1])
plt.show()

```

E. METADATA WIPING AND TRAIL OBFUSCATION EXPERIMENTS

In this experiment target's user used his PC for daily activities.

- The target machine is Windows 7 and the attacker machine is Kali-Linux. Both operating systems share the same network. Windows IP address is 10.0.0.10 and Kali-Linux IP address is 10.0.0.5. Wireshark ran on the host OS for capturing intermediary traffic between two machines.
- In the Kali-Linux to start Metasploit, the PostgreSQL service is started with `service PostgreSQL start` and the `ss -ant` command is used to check the state of the PostgreSQL. Next initialize the database by using `msfdb init` command. Next start metasploit with the `msfconsole` command. Simultaneously, nmap search is used to detect the OS version, the IP address, and the open ports of the target OS.
- On the Windows side, user created a test file (C:\FP.txt). Initial file name was "forensics Project Test File." The file name is changed to "FP.txt" for easy access the file from "msfconsole" during the exploit.
- Using exploit `ms15_100_mcl_exe` requires the user to open a purposely-crafted Windows Media Player list. Then the payload (.mcl) initiates a reverse TCP connection from target machine to attacker machine using crafted `mcl.exe` on port 4444 (Figure 12).

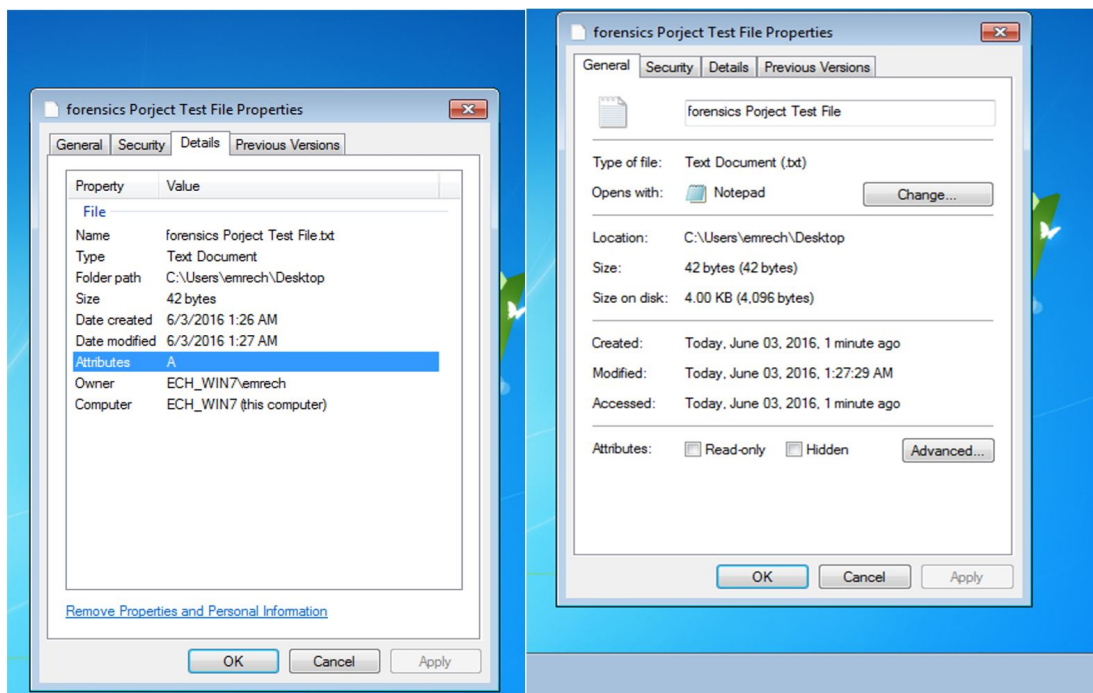


Figure 12. Test File Timestamps

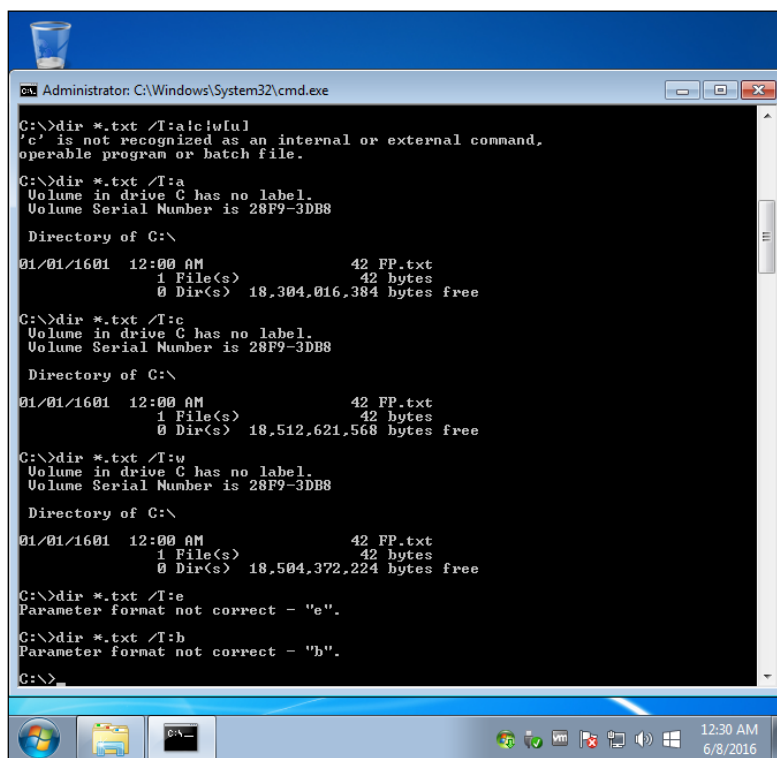


Figure 13. Result of Timestamp Change with timestamp

F. EVENT LOG MANIPULATION

After deleting or changing timestamps, an attacker can delete evidence of their presence on the system. For achieving this Metasploit provides “clearev” tool.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Beal, V. (n.d.). Encryption. Webopedia. Retrieved October 29, 2018, from <https://www.webopedia.com/TERM/E/encryption.html>
- Bender, W., Gruhl, D., Morimoto, N., & Lu, A. (2010). Techniques for data hiding. *IBM Systems Journal*, 35(3), 313–336. doi:10.1147/sj.353.0313
- Bergel, H. (2007). Hiding data, forensics and anti-forensics. *Communications of the ACM*, 15–20. doi:10.1145/1232743.1232761
- Böhme, R., & Kirchner, M. (2013). *Counter-forensics: Attacking image forensics*. New York, NY: Springer.
- Boneh, D., Sahai, A., & Waters, B. (2011). Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference* (pp. 253–273). Berlin Heidelberg: Springer.
- Burdach, M. (2006). Physical memory forensics. Retrieved from <https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Burdach.pdf>
- Conlan, K., Baggili, I., & Breitingner, F. (2016). Anti-forensics: Furthering digital forensic science through a new extended, granular taxonomy. *Digital Investigation* (18), 66–75.
- Counterfeit. (n.d.). Retrieved October 29, 2018, from <https://www.dictionary.com/browse/counterfeit>
- Foster, J. C., & Liu, V. (2005). Catch me, if you can. Retrieved May 5, 2017, from <https://www.bishopfox.com/resources/tools/other-free-tools/mafia/>
- Garfinkel, S. (2007). Anti-forensics: Techniques, detection and countermeasures. *2nd International Conference on Information Warfare and Security (ICIS)* (pp. 77–84).
- Garfinkel, S., & Shelat, A. (2003). Remembrance of data passed: A study of disk sanitization practices. *IEEE Security & Privacy*, 99(1), 17–27.
- Harris, R. (2006). Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. *Digital Investigation*, 3, 44–49. <https://dx.doi.org/10.1016/j.diin.2006.06.005>
- Hoglund, G., & Butler, J. (2006). *Rootkits: Subverting the Windows kernel* (2nd ed.). Stoughton, MA: Addison Wesley Professional.

- Huebner, E., Bern, D., & Wee, C. K. (2006). Data hiding in the NTFS file system. *Digital Investigation*, 3(4), 211–226. <https://doi.org/10.1016/j.diin.2006.10.005>
- Idrix. (n.d.). VeraCrypt documentation. Retrieved September 12, 2018, from <https://www.veracrypt.fr/en/Documentation.html>
- Jahankhani, H., & Beqiri, E. (2010). *Handbook of electronic security and digital forensics*. Retrieved from https://books.google.com/books/about/Handbook_of_Electronic_Security_and_Digi.html?id=ZgpV6Rvw2FoC
- Jain, A., & Chhabra, G. S. (2014). Anti-forensics techniques: An analytical review. *Contemporary Computing (IC3)*, 7, 412–418. <https://ieeexplore.ieee.org/document/6897209/>
- Kedziora, M., Chow, Y.W., & Susilo, W. (2017). Defeating plausible deniability of VeraCrypt hidden operating systems. In Batten L., Kim D., Zhang X., Li G. (Eds) *Applications and techniques in information security*: Vol. 719. Communications in Computer and Information Science (pp. 3–13).
- Ext4. (n.d.). Retrieved May 11, 2017, from https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout
- Kissel, R., Scholl, M., Skolochenko, S., & Li, X. (2012). *Guidelines for media sanitization revision 1*. Gaithersburg, MD: National Institute of Standards and Technology (NIST).
- Lillis, D., Becker, B., O’Sullivan, T., & Scanlon, M. (2016). Current challenges and future research areas for digital forensic investigation. arXiv preprint arXiv:1604.03850.
- Liu, V., & Brown, F. (2006). Bleeding-edge anti-forensics. Presentation at InfoSec World.
- Microsoft. (2003, March 28). How NTFS works. Retrieved from [https://technet.microsoft.com/en-us/library/cc781134\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx)
- Mishra, M., Mishra, P., & Adhikary, M. C. (2014). Digital image data hiding techniques: A comparative study. arXiv preprint arXiv:1408.3564.
- Raghavan, S. (2013). Digital forensic research: Current state of art. *CSI Transactions on ICT*, 1(1), 91–114. <https://doi.org/10.1007/s40012-012-0008-7>
- Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Rev*, 5(1), 3–55.

- Singh, M., & Mahajan, G. (2016). Various approaches of video and image stenography: A review. *International Journal of Science and Research (IJSR)*, 5(6), 1547–1549. <http://dx.doi.org/10.21275/v5i6.NOV164537>
- Sparks, S., & Butler, J. (2005). “Shadow walker”: Raising the bar for rootkit detection. Retrieved from <http://blackhat.com/presentations/bh-usa-05/bh-us-05-sparks.pdf>
- Swanson, M., Stoller, L., & Carter, J. (1998). Making distributed shared memory simple, yet efficient. In *High-level Parallel Programming Models and Supportive Environments* (pp. 2–13).
- Trojnara, M. (2016, November 28). Stunnel. Retrieved from <https://www.stunnel.org/index.html>
- Yusoff, Y., Ismail, R., & Hassan, Z. (2011). Common Phases of Computer Forensics Investigation Models. *International Journal Computer Sciences Information Technologies*, 3(3), 17–31.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California