Identification of Clear Text Data Obfuscated Within Active File Slack

Claire V. Wills

Follow this and additional works at: https://jagworks.southalabama.edu/theses_diss

Part of the Data Science Commons, and the Information Security Commons
IDENTIFICATION OF CLEAR TEXT DATA OBFUSCATED WITHIN ACTIVE FILE SLACK

A Thesis

Submitted to the Graduate Faculty of the University of South Alabama in partial fulfillment of the requirements for the degree of

Master of Science in

Computer and Information Sciences

by

Claire V. Wills
B.S., University of South Alabama, 2020
August 2022
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF ACRONYMS</td>
<td>vi</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER I INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Research Questions and Goals</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Expected Outcomes</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Outline</td>
<td>5</td>
</tr>
<tr>
<td>CHAPTER II LITERATURE REVIEW</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Digital Forensics Overview</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Hard Disk Drives</td>
<td>11</td>
</tr>
<tr>
<td>2.3 NTFS File Systems</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Slack Space</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Data Carving</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Next Steps</td>
<td>17</td>
</tr>
<tr>
<td>CHAPTER III METHODOLOGY</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Research Approach</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Experimental Setup</td>
<td>19</td>
</tr>
<tr>
<td>3.3 Phase I</td>
<td>20</td>
</tr>
<tr>
<td>3.4 Phase II</td>
<td>21</td>
</tr>
<tr>
<td>3.5 Phase III</td>
<td>23</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Research Phases for Identification of Clear Text in Slack Space Experiment</td>
<td>4</td>
</tr>
<tr>
<td>2. Experiment Results for Clear Text Identification in Slack Space</td>
<td>26</td>
</tr>
<tr>
<td>3. Descriptive Statistics for Total Words, Found Words, and Actual Words</td>
<td>28</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Outline of Elements in a Digital Forensics Investigation</td>
<td>10</td>
</tr>
<tr>
<td>2. HDD Partitioning Model</td>
<td>11</td>
</tr>
<tr>
<td>3. Structure of NTFS Partition</td>
<td>13</td>
</tr>
<tr>
<td>4. Diagram of Slack Space</td>
<td>15</td>
</tr>
<tr>
<td>5. Flowchart for Identification of Clear Text in Slack Space</td>
<td>22</td>
</tr>
<tr>
<td>6. Found vs Actual Clear Text Difference in Results</td>
<td>27</td>
</tr>
<tr>
<td>7. Delta Percentage of Words Found</td>
<td>32</td>
</tr>
</tbody>
</table>
LIST OF ACRONYMS

JPEG ................................................................. Joint Photographic Experts Group
HDD .............................................................................................................Hard Disk Drive
OS .............................................................................................................Operating System
NTFS ...................................................................................... New Technology File System
EXTX ................................................................................................. Extended File System
IoT .............................................................................................................Internet of Things
SSD ............................................................................................................Solid State Drive
MBR ...................................................................................................... Master Boot Record
MFT .......................................................................................................... Master File Table
VCN .......................................................................................................Virtual Cluster Number
LCN .......................................................................................................Logical Cluster Number
RAM ......................................................................................................Random Access Memory
ASCII .............................................. American Standard Code for Information Interchange
ABSTRACT

Wills, Claire, V., M. S., University of South Alabama, August 2022. Identification of Clear Text Data Obfuscated Within Active File Slack. Chair of Committee: Michael E. Black, Ph.D.

Obfuscating text on a hard drive can be done by utilizing the slack space of files. Text can be inserted into the area between the end of the file data and the New Technology File System (NTFS) cluster (the smallest drive space allocated to a file) that in which the file is stored, the data is hidden from traditional methods of viewing. If the hard drive is large, how does a digital forensics expert know where to look to find text that has been obfuscated? Searching through a large hard drive could take up a substantial amount of time that the expert possibly could not justify. If the digital forensics expert lacks the knowledge on how to properly search a hard drive for obfuscated clear text using data carving concepts, how will the obfuscated clear text be located on the drive and identified? To address this, an algorithm was proposed and tested, which resulted in the successful identification of clear text data in slack space with a percentage average of 99.31% identified. This algorithm is a reliable form of slack space analysis which can be used in conjunction with other data extraction methods to see the full scope of evidence on a drive.
CHAPTER I
INTRODUCTION

A cybercriminal can use different methods to hide illegally sourced data. One of the most common used methods is a technique called obfuscation. Obfuscation is the process of hiding or obscuring the data to make it look like something else to the user of the system or the file system (Fernando, 2021). The method of obfuscation this research focused upon is hiding data by manually inserting clear text data in the space allocated to a file, specifically slack space. Slack space is the area located between where a file ends and where the last cluster, the smallest drive space allocated to a file, that stores the file ends (Alji & Chougdali, 2021). The method of manually placing text can go undetected from digital forensics experts because slack space is allocated to the file, but anything contained within the slack space is not a part of the file. Viewing a file through a file explorer will not show the contents of slack space, this is because the logical file system must be ignored to view slack space. If the logical file system is used to view a file with slack space, any text manually placed would go completely undetected as it would be in the undetected area the logical file system ignores. This study will help with the identified knowledge gap in data obfuscation techniques by providing knowledge for a technique that is widely known by forensics experts but not widely known to the average system user and providing a starting point to research further.
A branch of cybersecurity, digital forensics, focuses on recovering and analyzing digital evidence gathered from digital media (Flores et al., 2021). It is a field that is continuously changing and growing due to the increasing number and complexity of cybercrimes. Digital forensics investigators rely on tools and their knowledge to identify digital evidence. Multiple tools are utilized to automate the parsing and recovering of data by a digital forensics expert to then analyze and determine if the digital data can be used as evidence (Horsman, 2021). These tools can range from scraping metadata time stamps from joint photographic experts group files (JPEGS) to listing out discovered file headers in slack space. In the digital forensics field, there has been an increasing problem with analyzing data to gather evidence because of the lack of training of digital forensics experts, encrypted data, and format incompatibilities (Garfinkel, 2010; Horsman & Sunde, 2020). There is a large increase in the number of criminal investigations that involve digital media with varying amounts of complexity which is leading to a lower quality of digital forensic results due to the rapid changes in the field (Casey, 2019). In fact, quality standards in digital forensics is directly related to cost savings (Tully et al., 2020). Digital forensics certificates are given to those who meet the minimum requirements (80% or better passing score) to pass the certification exam (Vincze, 2016), which allows for certified experts in the field only having a basic knowledge of tools and techniques. The lack of standards and constant changing for digital forensic experts creates an unqualified workforce for the high demand (Horsman & Sunde, 2020; Vincze, 2016). To help address the knowledge gap in digital forensics, identifying methods a cybercriminal uses to take advantage of modern computers for illegal means is a way to determine where knowledge gaps lie.
1.1 Research Question and Goals

The goal of the research is to determine if an algorithm can be developed which could potentially lead to a triage tool where plain text intentionally stored in slack space can be identified throughout a hard disk drive (HDD). The main capability will be to allow digital forensics investigators to execute a thorough search of text data in the magnetic drive’s slack space and to identify clear text data to be analyzed further by the investigator. The tool could potentially help close the current knowledge gap that digital forensics investigators face by automating the tedious work of data carving slack space to find obfuscated plain text.

Without any current tools used specifically automating the identification of clear text data in slack space, crucial information relating to an investigation can be overlooked entirely. Any useful data not identified and pulled from a magnetic drive can hinder an investigation. The more tools available to digital forensics experts to automate the processes of collecting and analyzing potential evidentiary data, the higher the chance all potential evidence is located on the hard drive (Renaud et al., 2021). Modern hard drives have become exceedingly large over the years and take time to analyze. By automating the process of finding clear text in slack space throughout the drive, the investigator can focus on interpreting the text data instead of carefully pulling out all areas of slack space. This research takes place in three phases, as outlined in Table 1.

Phase I of this research involves the development and setup of an experimental environment. The experimental environment involves creating a script to create test data
and a control file. The control file will be created which will contain randomized files with clear text data inserted to the end of the file.

Table 1. Research Phases for Identification of Clear Text in Slack Space Experiment

<table>
<thead>
<tr>
<th>Phase</th>
<th>Goals</th>
</tr>
</thead>
</table>
| I     | • Determine baseline methods for storing clear text in slack space  
       | • Store clear text data in slack space without using file system |
| II    | • Demonstrate viability of large amounts of clear text data inserted into slack space  
       | • Create random test cases |
| III   | • Develop methodology to automate identification of clear text in slack space  
       | • Measure reliability of clear text based on control file  
       | • Further evaluation of results |

The reason for this control file is to compare randomized files containing non-clear text data, which are randomized strings of letters, numbers, and special characters, in slack space to files containing clear text in slack space. The control file will contain the words from the dictionary that were written in the randomized areas of file slack throughout the drive and will also list the sector locations of each word. To achieve the goals of providing a quicker and more efficient method of identifying readable text data in magnetic drives the research question leading into Phase II is whether clear, readable text data be identified in the slack space of a magnetic drive.
Phase II demonstrates the experiment of placing clear text data in slack space of randomized files throughout the drive and testing the proposed algorithm. The proposed algorithm will be a bash script written with the purpose of extracting and identifying clear text from binary data. A proposed algorithm will be tested and evaluated to determine if clear text in slack space can be identified, possibly resulting in a triage tool. The results of the experiment will be evaluated against the control file created in Phase 1. The comparison of the files and the following analysis will determine the success of the algorithm.

Phase III focuses on the analysis of the results of the identification of clear text data experiment. The recoverability of clear text data will be measured by the amount of clear text recovered as compared to the amount of data stored as clear text in file slack based on the control file created in Phase II. Further evaluation will be conducted in which an analysis report consisting of a statistical analysis will provided upon review.

### 1.2 Expected Outcomes

The expectation of the researcher is to identify 95.00% of clear text data obfuscated in slack space. If the expectation is met, then a triage tool for identification of slack space could be developed from this research and expand upon the digital forensic knowledge base. If either of these outcomes are met, further study of this approach will be supported.

### 1.3 Outline
The remaining chapters of this thesis are organized as follows. Chapter II is a review of relevant literature including any previous research on clear text data in slack space, as well as providing context to all relevant technologies and processes. Chapter III covers a high-level overview of the research model and hypotheses used to achieve the research goal. Chapter III will also discuss the scripts that will be created to test the proposed algorithm that will find clear text data obfuscated in magnetic drives. Chapter IV outlines the results of the experiment conducted. Chapter V will discuss the analysis of the results from Chapter IV and conclude the research.
CHAPTER II
LITERATURE REVIEW

Digital forensics investigators rely on tools to be able to procure evidence from computers without altering the original drive. Digital forensics investigators face challenges with increasing complexity of digital media, issues of accessibility (encryption), technology rapidly changing, and increasing amounts of digital data to determine if the data is relevant evidence (Sunde & Dror, 2021). When data is obfuscated, there is an additional layer of complexity added to the investigation. No process, tool, technique or digital forensics investigator can guarantee results that are one hundred percent accurate, which highlights the importance of processes which will help alleviate some time to recheck results (Horsman & Sunde, 2020). Time constraints is a problem in each case that a digital forensics investigator works, which makes speed and thoroughness top priorities when searching a drive for digital evidence (Caviglione et al., 2017). The digital forensics investigator cannot examine every file on the drive which makes any digital forensics tools that automate tedious processes valuable (Horsman & Sunde, 2020). In this section, an overview of digital forensics, hard disk drive, NTFS file systems, the concept of slack space, and data carving, as well as information on digital forensics tools will be explained. The goal of this chapter is to provide an understanding
into where this research starts as well as how an algorithm, which will search for clear
text data that has been obfuscated within the hard drive, could be developed.

2.1 Digital Forensics Overview

Digital evidence is becoming more prevalent in investigations and continuing to
grow because of increasing popularity of smartphones and IoT (Internet of Things)
devices being connected to the Internet (Sunde & Dror, 2021). The quality of digital
forensic investigation results are decreasing, and with new ways to exploit digital media
always increasing makes cybercrime less comprehensible (Casey, 2019). Karie et al
(2019) states that digital forensics is a forensic science using proven methods to collect,
validate, preserve, interpret, present, identify, analyze, and document digital evidentiary
data. This type of forensics collecting is done by a digital forensics expert who uses their
expertise and specialized forensic tools to gather digital evidence from all digital devices
that were a part of the crime or could be of use in the case (Khalaf & Varol, 2019).

During the preservation, collection, validation, and identification phases, the
digital forensics expert is focused on collecting and sorting through data to determine
what is useful and what is not to the investigation (Khalaf & Varol, 2019). Using
specialized forensic tools, the digital forensics expert will make a bit for bit copy of the
original drive that was taken as evidence (Ferreira et al., 2020). The original drive needs
to remain unchanged to prove that the evidence collected off the copy of hard drive was
in fact on the original drive from the beginning (Warbhe et al., 2016). This process is
done by using the forensic image of the drive to find any relevant evidence (Khalaf &
Varol, 2019). To be able to prove the investigator did not tamper with the original drive,
a cryptographic hash function such as a MD5 or SHA256, which is an algorithm which takes a group of data and returns a fixed-sized hash, is used (Kishore & Kapoor, 2014). Hashing is using a mathematical algorithm which converts data into a fixed length of cryptographic randomized letters and numbers (Aggarwal et al., 2021). After making a forensic copy and hashing the original drive, the digital forensics investigator can show the forensic copy hash matches the original drive hash. The reason for this is to demonstrate evidence was not tampered with on the drive. By hashing the original drive and the forensic copy, an investigator search the forensic image to find any potential evidence without affecting the security, authenticity, and integrity of the data being presented in court (Michail et al., 2016).

Analysis, interpretation, documentation, and presentation rely heavily on the validity and authenticity of the data being presented in court (Silvarajoo et al., 2021). Analysis of the drive depends greatly on the knowledge of the digital forensics expert because of the major technical expertise needed when analyzing data (Adamu et al., 2020). Specialized tools are used during these phases to analyze data further to find any relevant evidence to the case (Fernando, 2021; Tomer et al., 2017). When documenting and presenting data, it is especially important to demonstrate that all evidence maintained integrity and authenticity as well as what processes, tools, operating system, and software was used throughout the evidence gathering phases (Owen & Thomas, 2011; Varol & Sönmez, 2017). The complexity involved in a digital forensics’ investigation can be seen in Figure 1. The stage this research focusses on is the data examination phase. This phase is a part of the examination phase due to the data hiding element of clear, readable text that has been manually placed by the user of the system in file slack which obfuscates the
data by bypassing the logical file system. Data hiding, or obfuscating, involves the large storage capacity of hard disk drives, time constraints, and lack of resources of forensic investigations to prevent the digital forensics investigator from identifying potential digital evidence (Wani et al., 2020).

Figure 1. Outline of Elements in a Digital Forensics Investigation (Horsman, 2021).

A problem in digital forensics today is known as “tool limitation”. Tool limitation is based on the knowledge gap of the digital forensic investigator and the limits of the tool itself (Fernando, 2021). This issue is essentially the Schrödinger’s cat of digital forensics. If evidence is not recovered from the electronic device due to lack of
knowledge of the digital forensics expert paired with tool limitations, an expert will not
know for sure all evidence has been found on the device (Horsman, 2018).

2.2 Hard Disk Drives

Hard disk drives are a form of non-volatile storage which is extremely common
(Mashhadi et al., 2018; Shahrad et al., 2018). What makes HDDs the more popular
choice over other forms of data storage is the vast amount of data that needs to be stored
and how HDDs are a less expensive option in comparison to solid state drives (SSD)
(Bennato et al., 2021; Li et al., 2019). In the past two decades there has been a rapid
increase in the advancement of hard disk drive technology (Gao, 2018). A hard disk drive
stores and retrieves data through the use of magnetic disks rapidly rotating (Chaudhary &
Kansal, 2015). The smallest physical unit of storage on a hard drive is called a sector and
is usually 512 bytes (Hao et al., 2019). As shown in Figure 2, the HDD partitioning
model contains a master boot record (MBR). The MBR contains the boot code for the
drive and the partition table information. Following the MBR are the partitions of the
drive (Gruhn, 2017). At least one primary partition needs to exist on the drive.

![Figure 2. HDD Partitioning Model (Bozic et al., 2017).](image-url)
An HDD using an MBR scheme, which this research will use, contains up to four partitions. If you need more than four partitions, one of the partitions can be made into an extended partition. An extended partition acts as a wrapper to hold other partitions created on the drive. Each partition on the drive is formatted with a file system. NTFS and extended file systems (EXTX) are the most common types due to being used in Windows and Linux operating systems.

### 2.3 NTFS File Systems

Operating systems have their own logical file system, and different file systems have different logical organization structures (Zhang et al., 2020). NTFS currently the Windows Operating System logical file which stores and manages the important data such as file allocation (Kai et al., 2010). Every object on the NTFS file system is a file (Huebner et al., 2006). The smallest logical unit in the NTFS file system is called a cluster (Oh et al., 2020). Cluster size can be set by the user and contain sectors, which are always 512 bytes (Zhu & Ma, 2018).

A NTFS partition contains several areas as seen in Figure 3 (Zhang et al., 2020). The areas are the boot, master file table (MFT), file storage, MFT Mirror, and another file storage space. The first sector of the NTFS file system is called the partition boot sector which contains information for the partition as well as the starting cluster of the MFT. Following the MFT is where all the nonresident files are stored. The last area is the MFT Mirror.
The first 12.5% of space is reserved for the MFT in an NTFS file system (Karresand et al., 2020). The MFT, is an indexing system for all files in the partition. Each file in the MFT is given a corresponding MFT entry, also known as a record. The size of the MFT record is 1024 bytes. The MFT Mirror is a copy of the MFT and is in another location on the drive to be able to recover the MFT if the MFT becomes corrupted or deleted from the drive.

Each record lists several file attributes which follow the MFT header. These attributes include the resident or nonresident attribute, data attribute, and several other attributes that contain information about the file and where it is stored on the drive. One important piece of information the MFT record metadata contains is the cluster numbers in which the file is stored (Galhuber & Luh, 2021). If a file has a nonresident attribute, the file is stored in a cluster run contained within the extent space, which is a consecutive run of clusters. If a file has a resident attribute, the file’s size is less than the MFT record (1024 bytes) and is stored within the MFT record itself. The MFT record does not have slack space. The first cluster number listed in the cluster run is known as the virtual cluster number (VCN). The logical cluster number (LCN), is the offset of the cluster which is the number of clusters the file is stored in. For example, if cluster numbers 10002, 10003, and 10004 are the logical clusters that a file is stored in, the VCN would be 10002 and the LCN would be 3.
The NTFS metadata files outline the layout of the file system (Huebner et al., 2006). These metadata files which are used to organize metadata and file content are also known as attributes (Palmbach & Breitinger, 2020). The first 16 attributes are defined by Microsoft, leaving 16 through 23 unallocated and extended attributes starting with 24. Looking at specific record numbers using SleuthKit the metadata of the file, including the file size, VCN (virtual cluster number), LCN (logical cluster number), and whether the file is resident or non-resident can be utilized (Cho, 2015).

### 2.4 Slack Space

To store a file on an HDD, a file larger than 1024 bytes on a partition formatted with a NTFS file system is stored in a group of clusters. If the file does not fill up the space in the last cluster storing the file, there is space leftover. That space is known as slack space (Alji & Chougdali, 2021). Volume slack and file slack are two of the main types of slack space. Volume slack is what is between the file system and the end of the partition the file system is on (Galhuber & Luh, 2021). Since the user can set the size of the cluster and the size of the partition, sometimes those numbers will not divide equally into each other which will cause remaining space to be unallocated, thus creating volume slack.

Slack space can be split up further into two different parts, which are file slack and random-access memory (RAM) slack. The space that is between the end of the file and where the last sector the file is stored in ends is RAM slack as seen in Figure 4. File slack, which is what this research focusses on, refers to the beginning of the following...
sector to the end last allocated sector within the cluster as seen in Figure 4. Figure 4 shows the difference in RAM slack and file slack which together make up slack space.

Figure 4. Diagram of Slack Space.

Since slack space is seen as allocated space on the file system, any data manually inserted in slack space is completely ignored by the file system (Srinivasan et al., 2017). This means that adding data to the file slack will not change the hash of the file because it is not considered part of the file anymore even though it is still allocated to the file. Clear text data inserted in slack space is automatically obfuscated from the file system, meaning none of the file system tools will recognize this area of the file as data (Bhat et al., 2021). When a user wants to get rid of the data in slack space, the user of the system needs to overwrite the file with a larger file than before.

Traditionally, slack space contains parts of deleted files (Mechtly et al., 2019). When a user deletes a file and overwrites it with another file, the parts of the file that remain which were not overwritten will stay in slack space. Readable clear text data should not be found within slack space due to files being binary. If clear text data is found, the text should be investigated further.


2.5 Data Carving

Data carving is a technique that pulls out a file header or footer belonging to a deleted file from a raw disk image while ignoring the logical file system (Ge et al., 2021). Data carving techniques are used to analyze space of a hard disk drive during a digital investigation (Ravi et al., 2016). Data carving ignores the logical file system and scans for known file headers at 512 byte offset (Gladyshev & James, 2017). There are several data carving techniques, file header based, metadata file carving based, file structure carving based, and footer carving based.

File signature based is identifying the header of a file in unallocated space. A unique identifier, file header, will associate the file with what type of file it is, whether it be jpeg, text, or any other type (Alshammary & Hadi, 2016). File headers are found at the beginning of a file making identification of where the start of a file is using data carving a simpler process. Header-footer carving based is like file header based except it also is identifying the tail of the file, which means that any space on the drive between bother the header and the footer can contain the contents found in the file. Footers are not usually found due to file fragmentation (Alherbawi et al., 2013). To recover a fragmented file, a file structure carving based technique needs to be used (Alshammary & Hadi, 2016). Metadata file carving is based around the metadata given by the file system to determine the file structure (Alshammary & Hadi, 2016). Data carving only addresses file signatures but does not address when a user places clear text in slack instead of deleting files from the system. In this paper, the carving technique used is not based around the headers or footers of files, but slack space will still need to be carved because the file system needing to be ignored to find clear text.
2.6 Next Steps

Understanding NTFS file system architecture on HDD partitioning structure leads to finding areas ignored by the file system in which clear text data can be obfuscated. The next steps involve the designing the controlled research experiment. Followed by testing the experiment and producing results. Analyzing the results metrics which relates to the research problem. Lastly, interpreting the results of the experiment and forming conclusions.
CHAPTER III

METHODOLOGY

An experimental environment using Ubuntu 20.04 OS was setup. Following the environment setup, a dictionary file containing 194,381 words excluding capital letters and special characters was created. A method of storing clear text data in slack space had been developed to create control data by generating 1000 unique files, which had strings inserted into slack space either found in the dictionary file (clear text), or variable strings of letters and numbers (non-clear text). A control file, in which the string of clear text or non-clear text and the sector location, was also generated which was created for testing the automation of identification of obfuscated clear text data. The control file was compared to the results produced and the bash script. The proposed algorithm was tested and compared to the control file resulting in further evaluation of results being conducted.

3.1 Research Approach

The researcher’s goal was to test a proof-of-concept algorithm that could aid in the development of a triage tool capable of identifying clear text data in slack space. Once a clear method of producing the data and control file was determined which created a test case, testing of the proposed algorithm was done and results were analyzed. For the experiment, the amount of clear text data identified in slack space determined the
precision of the algorithm. Non-clear text data, which was randomized strings of text not found in the dictionary file, was also inserted in slack space for further evaluation of the precision of the proposed algorithm. Success of the algorithm was quantified by the amount of obfuscated clear text data that was identified in slack space compared to the amount of clear text data stored in slack space by the control script. For the algorithm to have been considered successful, 95.00% of clear text stored in slack space needed to be identified. This percentage was chosen because the smaller the error amount for a forensics tool, the less time and effort is needed to verify results by the digital forensics expert, thus the deciding success factor was quantified at 95.00%. The metric was determined by the precision of the algorithm, based on how many words were recovered by the algorithm out of the total amount of clear text words that were randomly inserted into slack space by the control script, measured by the data from the control file.

3.2 Experimental Setup

A virtual environment was used with Ubuntu 20.04 LTS OS. Virtual Box was used as the virtualization software. A 1 GB secondary hard disk (dev/sdb), used for storing the files with clear text in slack space, was formatted with a NTFS file system (dev/sdb1) with a set cluster size of 4096 bytes. Supporting software, mkfs, fdisk, and vi editor was installed by the OS and stored on the boot drive (/dev/sda1) along with SleuthKit. Storing all tools on the boot drive preserved the tools when the secondary drive was reformatted between each experiment. Two bash scripts (create_data.sh and scan_slack.sh) were also stored on the boot drive. Create_data.sh inserts clear text data into the slack space of a randomized number between 1 and 1000 files consisting of
random data on the secondary drive (/dev/sdb1). The clusters were set to 4096 bytes in size while files were limited at 2048 bytes in size to fit into 4096-byte size clusters which left 2048 bytes for slack space. The remaining 2048 bytes (slack space) was where the random clear text word taken from the dictionary file (english.txt) was written to a random position within this area. Additionally, a variable American Standard Code for Information Interchange (ASCII) string was written to a random position in slack to create non-clear text data. The clear text and non-clear text data were never written to the same file slack. The non-clear text data was randomly written into slack space for verifying whether the proposed algorithm identified clear text from non-clear text data. For verification, the control file created from create_data.sh contained the location in slack space (as a sector) and the word or variable string written into that sector in slack space. Scan_slack.sh was the script for the proposed algorithm being evaluated.

3.3 Phase I

Phase I was composed of developing a script to produce control data and a control file (control.txt) to compare to the data the proposed algorithm identified in slack space which compared the data stored in slack to the data identified by the proposed algorithm. One thousand 4096-byte size files limited to 2048 bytes, by setting the file size using the fls command, were written to the secondary drive (/dev/sdb1). The control script (create_data.sh) then inserted random words from the dictionary file (english.txt) in random locations into the last 2048 bytes of a file and inserts non-clear text data which are randomized strings of text data not found in the dictionary file to a different file created by the control script. The random words in slack space along with the sector
location of the word are written to the control file for comparisons to the results of the proposed algorithm tested in Phase II.

### 3.4 Phase II

Phase II involved testing the proposed algorithm (*scan_slack.sh*) for the viability of identifying clear text data in slack space on a hard disk drive. To do this, a method of extracting slack space using a series of commands producing an output of all slack space to be compared to a dictionary file was devised. Once all data in slack space was exported to a file (*slackspace.dat*), the algorithm read the file line by line to validate whether a word from the dictionary file was identified. If the word was found, it was added to a file (*found.txt*) to be used for comparison against the control file. The algorithm focused solely on extracting clear text data from slack space using a bash script to locate any clear text data. Once the clear, readable text found on the HDD had been identified, the digital forensics investigator determines whether the data in slack space was pertinent evidence for the investigation. Phase II was successful in testing the proposed algorithm.

The bash script, which was developed, addressed the research problem by conducting the following tasks as shown in Figure 5:

1. Obtained a disk image.
2. Extrapolated slack space.
3. Converted slack space output from non-readable text to clear readable string data and saved to a file.
4. Compared each line of the file to a dictionary file.
5. If a word is identified as clear text in slack space, it was saved to a file for further analysis.

6. If non-clear text is identified, it was either unreadable patterns or words not found in the dictionary file.

Figure 5. Flowchart for Identification of Clear Text in Slack Space.
3.5 Phase III

Phase III consisted of analyzing the results of the experiment conducted in Phase II. The goal included measuring the identified clear text data of the proposed algorithm based on the control file created in Phase I by the amount of clear text data recovered from slack space by the proposed algorithm in Phase II. The algorithm was considered successful if $\geq 95.00\%$ of clear text was identified in slack space. A set of descriptive statistics was used on the data collected and was used in the analysis phase which determined the success of the proposed algorithm.
CHAPTER IV
EXPERIMENT AND RESULTS

After the experiment was run ten times, the results were recorded which included the words written to slack space from the control script (total words) and the clear text words found by the proposed algorithm (found words). After the analysis was conducted the recall and precision were recorded to reflect the error margin of the proposed algorithm.

4.1 Initial Results

A total of ten control and found files for each of the ten runs of the experiment were saved for analysis. Output of all data in slack space converted to strings (slackspace.dat), was also saved. The number of non-clear text data, clear text words from the control script, and found text identified by the algorithm was recorded after each run of the experiment. Table 2 contains five separate columns, the first column (run num) being the experiment number. Columns two (total words) and three (found words) contained the word count for the total words generated from the control script and written to the control file and the found words which was the word count found file generated by the proposed algorithm. Column four (extra words) was the extra words identified by the algorithm, but not part of the original words from the control file generated from the control script. Column five (missed words) was the number of words that were not found.
by the algorithm but created by the control script and found in the control file. Column six (actual words) was the actual words found by the algorithm after the extra words and missed words were accounted for. The results of each iteration of the experiment can be found in Table 2 along with the average of all ten experiment runs.

The initial data evaluation shows that the algorithm was successful by finding 99.41% of words generated by the control script as compared to the found file after taking the average of all the percentages for each run, but verification that each word was successfully found was needed. The data also presents a 0.59% margin of error which will be further explored. The recall percentage from Table 2, which is calculated by taking the actual words identified by the algorithm, dividing it by the number of clear text words in the control file (total words) and multiplying by one hundred.

Precision percentage from Table 2, which is calculated by dividing the actual words by the number of words initially found by the proposed algorithm (found words) and multiplying by one hundred. The percentages identified how successful the proposed algorithm was because each experiment run had a recall percentage over 95.00%. The precision percentage was 100.00% for most of the experiment runs, except for the runs that identified extra words.
Table 2. Experiment Results for Clear Text Identification in Slack Space

<table>
<thead>
<tr>
<th>Run Num</th>
<th>Total Words</th>
<th>Found Words</th>
<th>Extra Words</th>
<th>Missed Words</th>
<th>Actual Words</th>
<th>Recall Percent</th>
<th>Precision Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>493.00</td>
<td>493.00</td>
<td>2.00</td>
<td>2.00</td>
<td>491.00</td>
<td>99.59%</td>
<td>99.59%</td>
</tr>
<tr>
<td>2</td>
<td>515.00</td>
<td>511.00</td>
<td>0.00</td>
<td>4.00</td>
<td>511.00</td>
<td>99.22%</td>
<td>100.00%</td>
</tr>
<tr>
<td>3</td>
<td>504.00</td>
<td>503.00</td>
<td>0.00</td>
<td>1.00</td>
<td>503.00</td>
<td>99.80%</td>
<td>100.00%</td>
</tr>
<tr>
<td>4</td>
<td>481.00</td>
<td>476.00</td>
<td>1.00</td>
<td>6.00</td>
<td>475.00</td>
<td>98.75%</td>
<td>99.79%</td>
</tr>
<tr>
<td>5</td>
<td>479.00</td>
<td>477.00</td>
<td>1.00</td>
<td>3.00</td>
<td>476.00</td>
<td>99.37%</td>
<td>99.79%</td>
</tr>
<tr>
<td>6</td>
<td>499.00</td>
<td>498.00</td>
<td>0.00</td>
<td>1.00</td>
<td>498.00</td>
<td>99.80%</td>
<td>100.00%</td>
</tr>
<tr>
<td>7</td>
<td>507.00</td>
<td>503.00</td>
<td>0.00</td>
<td>4.00</td>
<td>503.00</td>
<td>99.21%</td>
<td>100.00%</td>
</tr>
<tr>
<td>8</td>
<td>485.00</td>
<td>480.00</td>
<td>0.00</td>
<td>5.00</td>
<td>480.00</td>
<td>98.97%</td>
<td>100.00%</td>
</tr>
<tr>
<td>9</td>
<td>481.00</td>
<td>477.00</td>
<td>1.00</td>
<td>5.00</td>
<td>476.00</td>
<td>98.96%</td>
<td>99.79%</td>
</tr>
<tr>
<td>10</td>
<td>501.00</td>
<td>498.00</td>
<td>0.00</td>
<td>3.00</td>
<td>498.00</td>
<td>99.40%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Avg</td>
<td>494.50</td>
<td>491.60</td>
<td>0.50</td>
<td>3.40</td>
<td>491.10</td>
<td>99.31%</td>
<td>99.90%</td>
</tr>
</tbody>
</table>

4.2 Results

After the initial findings, a manual comparison was done on both the control.txt file and the found.txt file. This was done doing a side-by-side comparison of both files by running the sdiff command to determine if the words match exactly. For the first run of the experiment, the initial results showed that 493.00 found words out of 493.00 total words in slack space matched, but further examination shows that there was a word difference of two words between both files. The algorithm also identified two extra words in slack space which were not found in the in the total words. The results of these discrepancies were recorded on Table 2 in columns four and five along with the actual number of words found by the algorithm.

After manual examination of all data, there were four iterations in which extra words were identified by the algorithm but not found in the total words. To elaborate on experiment run one, the initial results showed that 493.00 out of 493.00 words in slack space were identified, but after the control file and found file were manually verified the
algorithm did not identify two words resulting in only identifying 491.00 out of 493.00. When the two extra words are removed from the found words 491.00 remains, which also accounted for the two words the algorithm did not identify. The algorithm presented four errors, but the discrepancies canceled each other out which resulted in two words not identified. Runs 4, 5, and 9 each had one extra word identified by the algorithm that was not found in the total words. In Figure 6, the bar graph shows the initial number of words that were found as compared the actual number of words that were found by the algorithm resulting in slight changes between the four iterations that picked up extra words in slack space. Once all discrepancies in each iteration were accounted for, as shown in Figure 6, the actual found words were recorded and used for further analysis.

Figure 6. Found vs Actual Clear Text Difference in Results.
Descriptive statistics, including minimum, maximum, range, sum, count, and mean, was done on all ten iterations for the number of total words, the number of found words, and the actual words. To extend the descriptive statistics further, standard deviation, standard error, sample variance, kurtosis, and skewness was also calculated using the control words, found words, and actual words for all ten iterations on Table 3. The delta percentage shows the difference between the actual words and the total words generated by the control script. An average delta percentage of 0.69% the difference between the actual words ad total words is negligible and showed a successful proposed algorithm after ten runs of the experiment.

Table 3. Descriptive Statistics for Total Words, Found Words, and Actual Words

<table>
<thead>
<tr>
<th></th>
<th>Total Words</th>
<th>Found Words</th>
<th>Actual Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>479.00</td>
<td>476.00</td>
<td>475.00</td>
</tr>
<tr>
<td>Max</td>
<td>515.00</td>
<td>511.00</td>
<td>511.00</td>
</tr>
<tr>
<td>Range</td>
<td>36.00</td>
<td>35.00</td>
<td>36.00</td>
</tr>
<tr>
<td>Sum</td>
<td>4945.00</td>
<td>4916.00</td>
<td>4911.00</td>
</tr>
<tr>
<td>Count</td>
<td>10.00</td>
<td>10.00</td>
<td>10.00</td>
</tr>
<tr>
<td>Mean</td>
<td>494.50</td>
<td>491.60</td>
<td>491.10</td>
</tr>
<tr>
<td>Stand. Dev.</td>
<td>12.59</td>
<td>13.01</td>
<td>13.39</td>
</tr>
<tr>
<td>Stand. Error</td>
<td>3.98</td>
<td>4.12</td>
<td>4.23</td>
</tr>
<tr>
<td>Sam. Var.</td>
<td>158.50</td>
<td>169.38</td>
<td>179.21</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-1.35</td>
<td>-1.70</td>
<td>-1.70</td>
</tr>
<tr>
<td>Skewness</td>
<td>0.16</td>
<td>-0.05</td>
<td>-0.03</td>
</tr>
</tbody>
</table>

The descriptive statistics shows the data is mostly clustered around the mean with a symmetrical skewness. The data has low standard deviation of 13.01 and 13.39 for both the found words and the actual words. Compared to the total words of 12.59, each
column remains consistent throughout. Which shows a very small margin of error for both the found words and the actual words due to similar results.
Further analysis of the data needed to be conducted to determine the cause of the errors of the proposed algorithm. It was determined that the proposed algorithm was successful in identifying clear text in active file slack.

5.1 Analysis

After the results were recorded, a statistical analysis was done on the data. The descriptive statistics show the algorithm had an average reliability percentage of 99.31% with an average of ± 3.50 words either not found or being an extra word recognized by the algorithm, thus providing extra data that was not relevant to clear text generated by the control script. The additional words that the algorithm identified that were not found in the control file are all found containing uppercase letters. After examination of the extra words found compared to the control file, it was determined that the non-clear text strings were words that could be found in the dictionary file, but they were not clear text generated by the control script. When the randomized non-clear text strings were inserted into slack space, some of those strings randomly written random letters in an order that resulted in a word found in the dictionary file. Only run four, five, and nine of the ten runs of the experiment recognized one extra word within the non-clear text data which
was a randomized string of ASCII characters. Run one recognized two extra words within the non-clear text data. The extra words found was a side effect of the experiment that was unexpected, but shows the algorithm identifies and verifies clear text regardless of case or characters surrounding the word. For example, in the first run of the experiment one of the words found in slack space but not found in the control file was MaKo. The word mako can be found in the dictionary file, but not in the control file. The words that were not identified by the algorithm were because of the translation of the blkls output (slackspace.dat). After the data was analyzed in the control file, found file, and slackspace.dat file using the sdiff command, the words not being found by the blkls command are smaller three letter words. These words are found in slack space when validating the location of the word by using the dd command.

The experiment was still considered a success due to the 99.31% percentage average. Figure 7 charts all ten runs of the experiment as well as the delta percentage associated with each of the ten runs. Figure 7 exhibits a visual view of the statistical analysis. The mean delta percentage (0.69%) was found in the centralized area of the box in Figure 7 with a maximum of 1.25% and a minimum of 0.20% which has a range of 1.05%. Figure 8 was a visual view of the word difference which was found by subtracting the actual number of words found by the number of words in the control file. The average word difference (3.50) was also plotted and can be seen in the middle of the box in Figure 8 along with the maximum of 6 and the minimum of 1 and a range of 5.
5.2 Conclusions

The proposed algorithm was successful in identifying clear text data in slack space based on the definition of a successful experiment as outlined in Chapter III. The success of the proposed algorithm was determined by identifying 95.00% or better of clear text words in slack space and by having a 99.31% average percentage the algorithm was considered successful. One unexpected result of the experiment was the identification of words found in the dictionary file regardless of placement next to other ASCII characters or case. When the extra words were identified, it skewed the initial results which made the algorithm appear 0.10% more successful than it was. The proposed algorithm could be used for identification of longer words or strings which could lead to quickly identifying malware obfuscated in slack space or to help identify clear text in slack space in a digital investigation.
5.3 Limitations

There are four limitations of the experiment. The first limitation involves being limited to only using NTFS file systems. NTFS is the default file system used in Window’s Operating System, which has about 86% of the market share, (Karresand et al., 2020). By only using one type of file system, it narrowed the scope of the experiment.

The second limitation for the experiment was other forms of data storage, such as SSDs (solid state drives), are not being used. HDD architecture allows for slack space and are still a popular choice for data storage due to being cheaper and having a larger capacity than SSDs (Brown et al., 2021). SSDs do not have slack because of the use of solid state memory (Alsayoud & Miri, 2019). SSDs store information on flash memory chips which means information is stored as electronic charges without moving parts coming into play (Kim et al., 2018).

Dictionary bias was the third limitation and played a factor in this experiment. An American-English dictionary located in the dictionary file (english.txt) was used when creating the clear text data allocated to randomized files. The dictionary did not contain special characters or uppercase letters, so those were excluded in the experiment. The proposed algorithm ran words against the dictionary file and only words contained in the specific dictionary were identified. This file was not an unabridged dictionary and was limited in size to 194,381 words.

The fourth limitation of this experiment was that only ASCII text characters were be recognized. ASCII is 1 byte sized numeric characters that range from 0 through 255 (Mukhopadhyay et al., 2017). Only alphabet ASCII characters were used in the experiment. This left out password combinations and usernames which typically involve
special characters and numbers. Clear, readable American English dictionary-based
text words were identified, leaving out any number patterns.

5.4 Future Work

The proposed algorithm was considered successful but can be expounded upon
which could become a triage tool for digital forensics experts to use to scan for any
information that does not belong in slack space. To expand the research, the location can
be added to the found file to provide results that will give the user a sector location on the
drive to find where the identified clear text resides. By adding a location to the found file,
the investigator would be able to go to the sector location of the drive to be able to
investigate the clear text identified. Research could also identify special characters and
numbers. The research presented only identifies clear text, but with the addition of
special characters and numbers being identified, passwords and credit card numbers that
a criminal could be obfuscating in slack space could also be found. If the research is
expanded with special characters and numbers, malware code embedded in slack space
could potentially be identified.
REFERENCES

https://doi.org/10.1109/ISDFS49300.2020.9116375


https://doi.org/10.1016/j.protcy.2013.12.165


https://doi.org/10.1109/AICCSA47632.2019.9035253


digital forensics: Reliability and biasability in digital forensics decision making.
https://doi.org/10.1016/j.fsidi.2021.301175

forensics. *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN),* 188–192.
https://doi.org/10.1109/IC3TSN.2017.8284474

standards for digital forensics: Learning from experience in england & wales.
https://doi.org/10.1016/j.fsidi.2020.200905

Varol, A., & Sönmez, Y. Ü. (2017). Review of evidence analysis and reporting phases in
https://doi.org/10.1109/UBMK.2017.8093563


https://doi.org/10.1016/S1361-3723(20)30030-0


APPENDIX

Appendix A: Scripts

`create_data.sh`

```bash
#!/bin/bash

count=1000
sudo umount /mnt
sudo mkfs.ntfs -c 4096 /dev/sdb1
sudo mount /dev/sdb1 /mnt
rm control.txt
rm -rf /mnt
for x in $(seq 1 $count)
do
dd if=/dev/random count=2 bs=1024 of=/mnt/file_$x
done

links=(`fls -u /dev/sdb1 | grep -i "file_" | cut -d"-" -f1 | cut -d" " -f2`)
sectors=()
lines=`wc -l english.txt | cut -d " " -f1`
for x in ${links[@]}
do
sectors+=(`istat /dev/sdb1 $x | tail -n1`)```
done
count=0
for x in \${sectors[@]}
do
echo $count
    count=\$(($count + 1))
location=`shuf -i 2048-4000 -n1`
random_location=`shuf -i 2048-4000 -n1`
word=\$(\$(sed -n \$(\$(shuf -i 1-$lines -n1)p english.txt) 1-\$lines -n1)p english.txt)
r=$(( $RANDOM % 2 ))
if [ $r -eq 0 ]; then
    echo "$word" | dd of=/dev/sdb1 bs=1 seek=$(\$(\$(x*4096+$location)) 1-\$location))
    echo -e "1 $word," $(\$(\$(x*4096+$location)) 1-\$location)) >> control.txt
else
    raw=$(openssl rand -base64 \`shuf -i 2-10 -n1\``)
    echo "$raw" | dd of=/dev/sdb1 bs=1 seek=$(\$(\$(x*4096+$location)) 1-\$location))
    echo -e "2 $raw," $(\$(\$(x*4096+$location)) 1-\$location)) >> control.txt
fi
done

scan_slack.sh

#!/bin/bash
rm -f found.txt
rm -f cleartext.txt
rm -f falsedata.txt
x=10
blkls -s /dev/sdb1 | strings > slackspace.dat
while read line
do
grep -x -i "$line" english.txt
if [ $? -eq 0 ]
then
  i=\$((i + 1))
  echo "$i $line" >> found.txt
fi
done < slackspace.dat
while read line
do
um=`(echo $line | cut -d " " -f1)`
if [ $num -eq 1 ]
then
  echo $num >> cleartext.txt
else
  echo $num >> falsedata.txt
fi
done < control.txt
foundnum=`wc -l < found.txt`
textnum=`wc -l < cleartext.txt`
falsenumber=`wc -l < falsedata.txt`
echo "Number of Files: $((textnum+falsenum))"

echo "Bogus Data: $falsenum"

echo "Control: $textnum"

echo "Found: $foundnum"

mkdir /home/ite/scripts/files/$x

cp control.txt /home/ite/scripts/files/$x/control$x.txt

cp slackspace.dat /home/ite/scripts/files/$x/slackspace$x.dat

cp found.txt /home/ite/scripts/files/$x/found$x.txt
BIOGRAPHICAL SKETCH

Name of Author: Claire V. Wills

Graduate and Undergraduate Schools Attended:

University of South Alabama, Mobile, Alabama

Degrees Awarded:

Bachelor of Science in Information Technology, 2020, Mobile, Alabama

Master of Science in Computer and Information Systems, 2022, Mobile, Alabama

Awards and Honors:

National Science Foundation Scholarship for Service, 2019