Insider Risk: Finding Sensitive Files in the Enterprise using a PC's Master File Table

Michael R. Lehrfeld
Department of Computing
East Tennessee State University
Johnson City, TN 37614
423-439-6952
lehrfeld@etsu.edu

Abstract

Individuals whom have legitimate access to network resources, trade secrets, or otherwise sensitive data as part of their daily functions are categorized as an Insider Risk. Insider Risk has been pushed into the public eye in recent years with the Edward Snowden leaks of 2013. Snowden had a business need to access the data he retrieved but the controls around how that data was used were insufficient to protect it. It is important to note is that an Insider Risk does not have to have malicious intent. Human error can cause a data beach just as easily as a hacker can. The problem this paper address is one where users have the proper access to the resources they need while at the same time enabling an organization to monitor where that data resides during its useful lifecycle (for example, if that data is copied to a USB drive or to a cloud storage platform). This paper documents a tool that can be used to inventory known sensitive files throughout an enterprise using a PCs Master File Table. The first step in preventing an Insider Risk from causing a breach is to have an accurate assessment of where the data is, then appropriate actions can be deployed if needed.

Introduction

Insider Risk (or an insider) is a broad and complex problem. In traditional cybersecurity, the hackers are attacking corporate resources through traditional vectors; phishing emails, web site vulnerabilities, USB phishing attacks, misconfiguration in software, or basic malware. Although we might not know who the attacker is, we have a firm understanding of where they are coming from and what they might be after. An Insider Risk is different. An Insider Risk can take many forms (figure 1).

Figure 1. Various manifestations of an Insider Risk

According to Verizon's Data Breach Report (Verizon, 2017), over the 42,000 incidents that were reported, 25% of them were found to have origins with an insider. In other words, someone that was given access to resources misused them causing a breach. Continuing with this trend, 2,500 of the total insider compromises are attributed to 'Miscellaneous Errors' (i.e. Human Error). To exacerbate the insider problem, Verizon reports that Insider Risk breaches often take orders of magnitude longer to discover than traditional, hacker based breaches. Often going undetected for months and many instances, years. The problem becomes one of how do organizations protect their resources from the very people that are supposed to have legitimate access to those assets. One methodology is to attempt to understand the motives behind why an insider causes harm to their employer or steals assets from them.

Goodenough and Decker (2008) investigate the applicability of cognitive neuroscience (the biological methods that influence thought and understanding) to the Insider Risk problem in their paper "Why do Good People Steal Intellectual Property". This domain strives to understand a link between a person's physiological nature of their brain and how a person reacts and thinks in various situations. The authors use an analogy of a song to illustrate how this applies to intellectual property theft. Playing that song or singing it does not omit the opportunity for another person to do the same thing. This is drastically different from physical theft where if a chair is stolen then no one else can use that chair. It is this key differentiation in the type of property stolen where the 'why' insiders steal. With the theft of the song, as is true with Intellectual Property (IP) theft, overuse of the IP does not cause the IP to go away. Admittedly, the IP valuation may decline because of overuse, but the IP can still be used. With the theft of the chair, the chair is gone and goes away. While the utility behind attempting to understand the motives of insiders is debatable, the problem that insiders pose to organization is still very tangible.

Tackling the idea of mitigating the threat posed by insiders by monitoring the databases where the IP is stored is a concept presented by Bertino and Ghinita (2011). In their work, Bertino and Ghinita propose a monitoring system for database management systems (DBMS) that overlays the database and uses anomalous activity detection to determine if a user's activities are malicious or not. They noted that it is ideal to monitor as closely to the IP as possible to get the best results for the anomaly detection engine. That is the reason for the tight coupling with a DBMS. Their results were promising and they were able to detect some data exfiltration cases that traditional firewalls might have missed or ignored. It is noted that many data exfiltration activities will piggyback upon normal network communication channels; many choosing to simply encrypt the data and forward it out to the Internet via HTTP port 443. Bertino and Ghinita proposed a simple architecture where all traffic to and from the DBMS routes through a secure gateway where analysis is performed. Their approach has a couple of challenges. The idea of a secure gateway introduces the potential for a single point of failure as well as a chokepoint for the DBMS. Secondly, and more crucial to insider risk, is that IP can reside in other locations than a DBMS. Within many organizations, IP is located on Microsoft SharePoint sites, file servers, and even stored on technicians workstations. This large and diverse surface makes detection difficult and a single point system only mitigating some of the IP protection needs. The work documented in this paper will not mitigate or prevent data leaving an organization. However, it will help to forensically identify where the data has been staged as well as provide for a data map of where the IP has been.

Anatomy of the Master File Table

The Master File Table (MFT) is a file that is contained on every NTFS file system. This file is named $MFT and is located in the root of the file system but hidden from users unless forensics tools are used (figure 1).

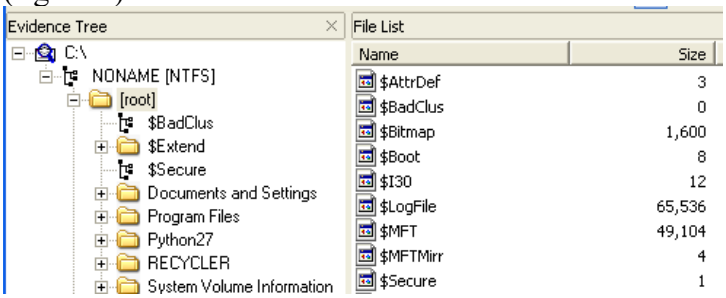


Figure 1. Master file table located in the root of the drive using AccessData's FTK Imager.

The MFT is a table of contents of sorts for all files and folders within a given file system. There is, at a minimum, one entry in the MFT for each file (Carvey, 2014). These entries include metadata about the file including: 1) file name, 2) file size, 3) file location on the disk, 4) data/time stamp information, 5) permissions, and 6) file content (when the file is smaller than ~900 bytes). As files and folders are created, entries in the MFT are also created. However, when files are deleted, they are marked for deletion (i.e. reuse) but not removed. When new files are added, depending upon the size of the MFT, they are either added to old (deleted) entries or to new entries. The reason for this functionality is the MFT is a high volume file that is accessed frequently. To increase the access times of the MFT, Microsoft has designed the MFT to reside in contiguous space on hard drives for increased performance. Because of this, the MFT is preallocated space larger than is needed. It is very common for an MFT to contain an abundance of deleted information – which is good for forensics.

The data structure of the MFT that contains file information is the $FILE_NAME attribute. Within this attribute is the metadata about a particular file. Table 1 is a listing of the $FILE_NAME attributes.

Offset	Length (bytes)	Description
0x00	8	Reference of parent directory
0x08	8	File creation time
0x10	8	File modification time
0x18	8	MFT modification time
0x20	8	File access time
0x28	8	Physical file size
0x30	8	Logical file size
0x38	4	Attributes (flags)
0x3C	4	Extended attributes
0x40	1	Length of file name
0x41	1	File namespace
0x42	Variable	File name

Table 1. $FILE_NAME attribute data structure (Polstra, 2016).

Using this information, we can start to directly process the MFT for interesting file names and start to build a map of where these files are currently located, or have been located in the past.

Before the MFT file can be processed by our script, it must be converted into another format using the attribute definitions from Table 1 and others defined by (Carrier, 2005). Figure 2 is an example of an MFT file using the forensics analysis tool FTK Imager. Notice the excessive noise in the file as it is not in a human readable format. Notice in figure 2 the sample file name "DEATHS1.DOC1". This is an example of the file name attribute (0x42). It is a short file name in 8.3 format that is commonly used for backward compatibility when applications cannot process long file names with spaces in them. The long file name is also stored in the MFT and can be seen as "Death Star Floor 1.docx". It would be difficult to search this type of file and provide meaningful output without pre-processing activities.

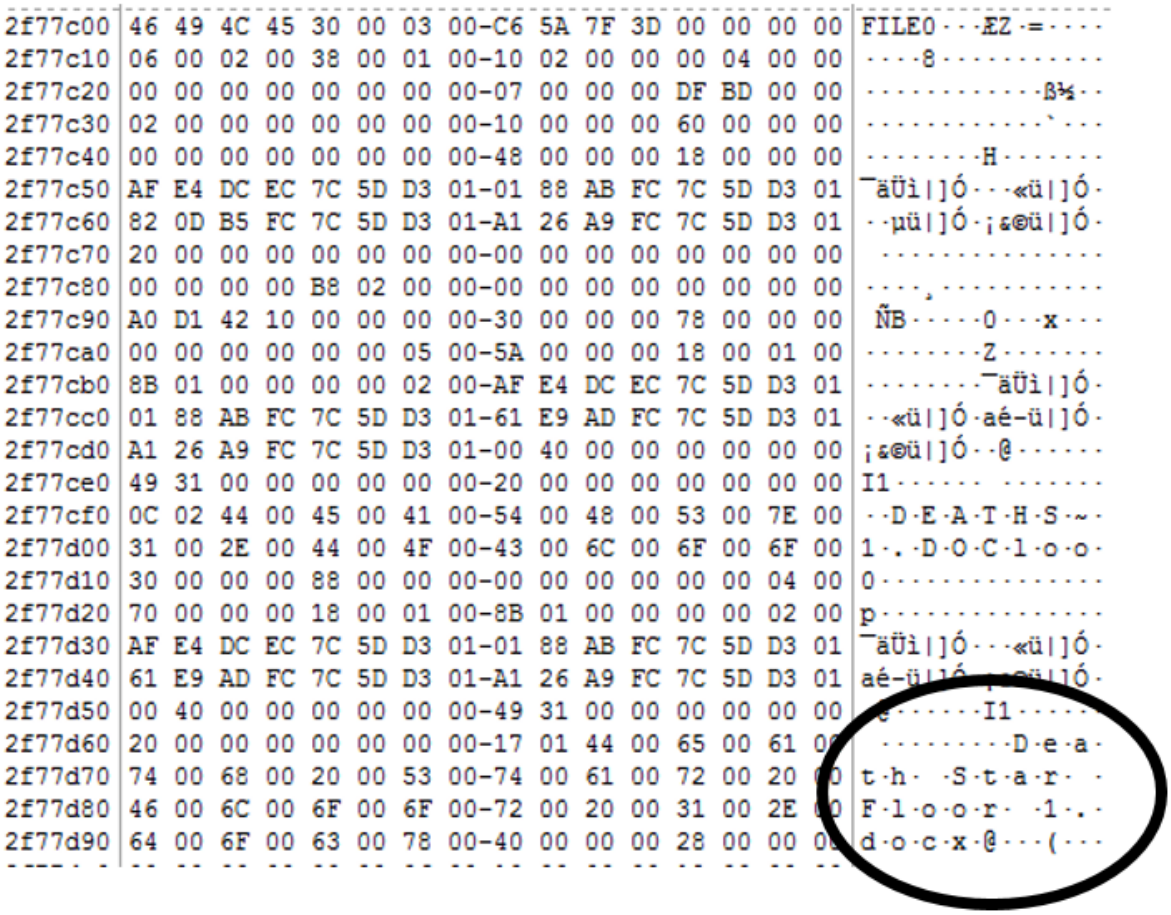
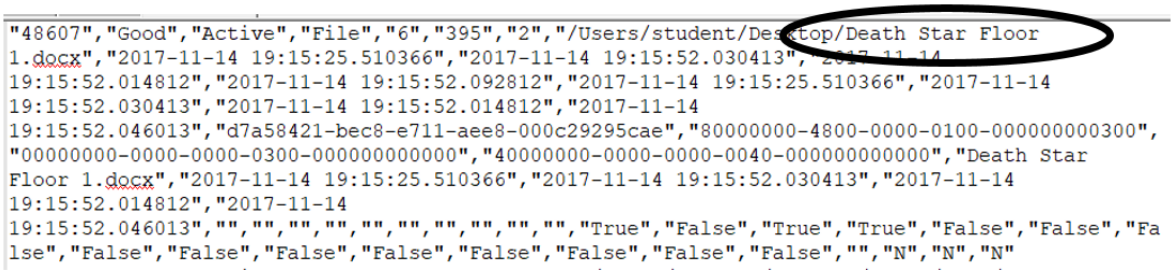


Figure 2. Raw example of a record in an $MFT viewed using a forensics analysis tool – FTK Imager.

Converting the MFT into a format which can be searched is done utilizing a third party script, analyzeMFT (dkovar, 2018). This python script takes a $MFT as an input and returns the MFT in a .csv format. The .csv format is easier to import into other python scripts and allows for simple searching of interesting files. Figure 3 is an excerpt from a processed $MFT that is in .csv format. After the MFT files have been collected and processed into .csv files, searching them for interesting information can commence.



```
"48607","Good","Active","File","6","395","2","/Users/student/Desktop/Death Star Floor
1.docx","2017-11-14 19:15:25.510366","2017-11-14 19:15:52.030413","2017-11-14
19:15:52.014812","2017-11-14 19:15:52.092812","2017-11-14 19:15:25.510366","2017-11-14
19:15:52.030413","2017-11-14 19:15:52.014812","2017-11-14
19:15:52.046013","d7a58421-bec8-e711-aee8-000c29295cae","80000000-4800-0000-0100-000000000300",
"00000000-0000-0000-0300-000000000000","40000000-0000-0000-0040-000000000000","Death Star
Floor 1.docx","2017-11-14 19:15:25.510366","2017-11-14 19:15:52.030413","2017-11-14
19:15:52.014812","2017-11-14
19:15:52.046013","","","","","","","","","","","True","False","True","True","False","False","Fa
lse","False","False","False","False","False","False","False","False","","N","N","N"
```

Figure 3. Example of a comma separated MFT record that has been extracted from $MFT and converted into a .csv file.

Searching for Protected Data

Searching through a .csv file using python is a relatively trivial task. The difficult work for an organization is the development and maintenance of a list of known files that contain IP that is being monitored. As the MFT file does not contain other, more detailed information about files, searching for file names (rather than user names or file contents) is limited. It is noted that files that are smaller than 900 bytes are stored entirely within the MFT data structure. However, no files of IP were found of that size, so that option is dismissed as a validation method for the tool. Figure 4 is the python script that searches file names and Figure 5 is sample output and the keywords to be searched.

```
import sys
import csv

with open(sys.argv[1], 'r') as terms:
    lines = terms.read().splitlines()

print("\n+--- Here are our search words ---+\n")
for each in lines:
    print(each)
print("\n+--- MFT from Computer ", sys.argv[2], "---+")
print ("\nSearch Term -- Record Num -- File Name")

with open(sys.argv[2], 'r') as f:
    readMFT = csv.reader(f, quotechar='"', delimiter=',', quoting=csv.QUOTE_ALL,
    skipinitialspace=True)
    for row in readMFT:
        row = [element.lower() for element in row]
        for term in lines:
            if [x for x in row if term in x]:
                print(term,  row[0], row[7])
```

Figure 4. Python 3 script that searches the parsed $MFT file.

```
C:\Users\lehrf\Documents\python\mft>python searchMFT.py keywords.txt parsedMFT.txt

+--- Here are our search words ---+

luke skywalker
star floor

+--- MFT from Computer  parsedMFT.txt ---+

Search Term -- Record Num -- File Name
star floor 48607 /users/student/desktop/death star floor 1.docx
star floor 48611 /users/student/documents/plans/~$ath star floor 2.docx
star floor 48619 /users/student/appdata/roaming/microsoft/office/recent/death star floor 2.lnk
star floor 48623 /users/student/documents/plans/death star floor 2.docx
```

Figure 5. Output from the IP search python script. Notice the search words and the outputted MFT entries.

The IP script can easily be modified to output the results to a file for further analysis. Increasing the utility of this script would be automating the processing of $MFT files and searching for known files that contain IP. When a new file location is identified, a notice can be triggered and delivered to an operator.

Applying the IP Script in the Enterprise

This tool was tested on a global network under diverse client conditions. Some of the challenges and successes of the tool will be discussed in this section in general terms to protect the subject's environment.

Challenges: This first challenge is not directly related to searching the MFT, but rather the collection of search terms. In large scale enterprises it is often difficulty to identify file names that need to be monitored. In this project, identifying file names was accomplished by locating data owners that had responsibility over various types of IP documents. From here, a list was generated that could be used to search over collected $MFTs.

Collecting $MFTs from endpoints can be another complication of this process. $MFT files are protected, hidden files that are located on the Windows root directory. Traditional methods of file copying will not be able to access and copy the $MFT. This causes the wide scale collection of $MFT files to be difficult. To accomplish this, specialized tools are required. Forensics tools are able to access these files, as well as advanced endpoint malware tools. This project utilized a specialized proprietary endpoint tool for the mass collection of the $MFT files. Free forensics tools like FTK Imager (FTK Imager, 2018) are also useful for collecting the $MFT. However, it is often more complex to capture global $MFT files using FTK Imager. Another method that could be used to capture the $MFT file is a PowerShell script. An incident response script has been published on github (https://github.com/n3l5/irFARTpull) that has the capability to copy $MFT files and has the ability to be automated (n135, 2018).

Successes: Searching for IP files on a large scale network is a complicated task. Having a mechanism where a logical map can be created that contains the location of IP files is helpful in identifying an insider risk. The first step was to create a baseline of where all the files currently reside. This allows the creation of a map of where files live or have previously been, and more importantly, when new locations appear.

Utilizing the initial baseline it is possible to quickly identify new machines that have IP downloaded to them. Creating alerts on the newly identified machines enables incident responders to investigate. Depending upon the alert timing, this advanced warning gives security teams a proactive edge into locating where IP is and how it moves within the corporate infrastructure. Creating an alert structure could be facilitated by an organizations security information and event management system (SIEM). The IP script can send logs to the SIEM where an IP dashboard would present any new locations found. Thus enabling security teams to proactively monitor any changes in IP. Figure 6 is an example of what a baseline would look like if IP locations were overlaid onto a map.

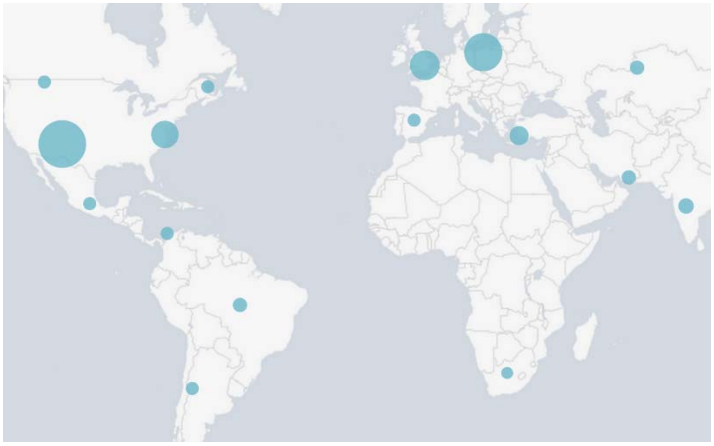


Figure 6. Example visualization based upon the found locations of IP on machines (Splunk Answers, 2018).

Conclusion and Future Work

The IP script was successful in what it was created to do – locate IP files across an enterprise. However, it was determined to be too narrow in scope as it required an exact file name match (or wild carding). The following areas of future work would have increased the capability of the IP script and potentially making it a more versatile tool in combating an insider.

A potential blind spot with searching $MFTs is the need for an exact match for the files names, or at a minimum, a partial match using wildcards. An upgrade to the current script would involve the use of fuzzy logic to aid in the pattern matching of the file names. A promising R package that utilizes the Damerau-Levenshtein distance algorithm, among others, to allow for fuzzy logic searching of the $MFT (Van der Loo, 2014). The Damerau-Levenshtein distance is an algorithm that calculates the minimum number of permutation to transform one word into another. This fuzzy logic would add to the capabilities of the script and account for minor changes in a files name. This type of expansion would aid in combating anti-forensics measures that an insider might use to mask their efforts.

Another area of expansion is the application of data science algorithms to analyze trends in the data. Using asset management within an organization, it would be possible to physically track the location of IP. This might provide for an interesting analysis of the spread of IP within a particular group. Additionally, it would be beneficial to map the physical location of IP within an organization. Knowing who has access to IP and their physical location are two distinct components of an insider risk program. The ability to combine both would greatly add to the capabilities to detect an insider within an organization.

References

Bertino, E., & Ghinita, G. (2011). *Towards mechanisms for detection and prevention of data exfiltration by insiders: keynote talk paper*. Paper presented at the Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, Hong Kong, China.

Carrier, B. (2005). File System Forensics. Upper Saddle River, NJ: Addison-Wesley Professional.

Carvey, H. (2014). Windows Forensics Analysis Toolkit (4th ed.). Waltham, MA USA: Syngress.
dkovar. (2018). analyzeMFT. Retrieved from https://github.com/dkovar/analyzeMFT

FTK Imager. (2018). FTK Imager. Retrieved from https://accessdata.com/product-download/ftk-imager-version-3.4.3

Goodenough, O. R., & Decker, G. J. (2009). *Why Do Good People Steal Intellectual Property?* Retrieved from Ashgate: https://ssrn.com/abstract=1518952

irFARTpull. (2018). Incident Response Artifact Pull. Retrieved from https://github.com/n3l5/irFARTpull

MSDN. (2018). Master File Table. Developer Network. Retrieved from https://msdn.microsoft.com/en-us/library/windows/desktop/aa365230(v=vs.85).aspx

Polstra, P. (2016). Windows Forensics: Pentester Academy.

Splunk Answers. (2018). GEOSTATS display bubbles on map instead of pie chart. Retrieved from https://answers.splunk.com/answers/221348/geostats-display-bubbles-on-map-instead-of-pie-cha.html

Van der Loo, M. P. (2014). The stringdist package for approximate string matching. The R Journal, 6(1), 111-122.

Verizon. (2017). Verizon data breach investigations report. 10th Edition. Retrieved from http://www.verizonenterprise.com/verizon-insights-lab/dbir/