Towards Proactive Forensic Evidentiary Collection

Clay Shields
Department of Computer Science
Georgetown University
Washington, D.C., 20057
clay@cs.georgetown.edu

Abstract

Forensic investigations have traditionally relied on data that exists as a by-product of normal operating system and application operation on a system following an incident. We propose a research agenda targeted at expanding the information available to an investigator in computing environments in which software can be installed on the target systems ahead of any incident. In these cases, information can be preserved proactively *and stored until needed for examination. In our first ongoing project, we are working to modify a file system to selectively recover disk blocks that are less likely to contain useful information when space is needed for a new file. In our second, we are keeping small amounts of information about files on a system that are deleted, copied, or modified. This allows us to perform certain types of investigations on files that are overwritten or otherwise missing from the system.*

1. Introduction

Computer forensics is historically a reactionary endeavor. Investigators respond to allegations of misuse or criminal activity once it has occurred and gather evidence to support or refute a hypothesis about what occurred on the system. They are dependent on the artifacts that are left on the system, either as a byproduct of normal operation or as part of the system logging. In many cases this evidence is sufficient to allow the investigators to reach a conclusion; in others it is not. The difference often lies in non-deterministic computer activity that can overwrite or otherwise destroy potentially valuable evidence. It is currently difficult or impossible to determine what evidence might remain and for how long.

In this paper we present a research agenda to make forensic evidence gathering proactive. We argue that in many environments, especially centrally administered networks under corporate or government control where the bulk of forensic investigations occur, it is possible to make changes to computer systems that will either preferentially preserve data with potential evidentiary value or will purposefully collect forensic data in advance of any activity which would trigger an investigation. We present two ongoing projects that are pursing this agenda.

In the first effort we are working towards modifying computer file systems so that deleted information that is more likely to be of forensic interest is less likely to be overwritten for new files. In a file system with these modifications, we maintain metadata about each deleted file, including the file owner and file type. When the operating system needs to reclaim disk blocks containing a deleted file, we use this metadata to select blocks that are less likely to be of interest to an examiner. We are currently working on a method of measuring the longevity of deleted files on a variety of operating and file systems using real-world file system traces. Once we are able to accurately measure the differences between file systems, we will modify a Linux file system to evaluate the effectiveness of this approach.

In the second project, we are developing a system that creates and stores information about files as they are deleted. Using information retrieval techniques, we create a *signature* for each deleted document which is stored in a database. A file signature contains metadata about the file along with one or more fingerprints of the file. A file fingerprint, which is typically just a few hundred bytes or less, is created based on the contents of the document so that it is robust against small edits to the file. This means that given a particular document we can probabilistically find all systems across a network that ever contained a copy of that document, even if the document is not in electronic form. This is useful for intrusion investigations in which a particular attacker tool is recovered, as all other systems that had that tool are easily identifiable. It is also useful in counterintelligence operations where recovery of a leaked document can be used to identify systems that contained that document for further investigation. Additionally, some

fingerprints store information about keywords that were in documents so an investigator can search across systems to find which contained documents matching a particular set of keywords. This can be used to determine which machines on a system require a full forensic examination, making better use of expensive investigative resources.

In the next section, we describe differences in the environments in which forensic response occurs and how those environments provide new opportunities for data collection. In Section 3. we describe a proactive approach involving file system modifications. In Section 4. we describe a proactive approach in which we use information retrieval techniques to record information that allows us to recognize or search files even if deleted or overwritten.

2. A Proactive Approach to Computer Forensics

The popular image of computer forensic examination is that of police or intelligence agencies seizing computers from a crime scene or intelligence target and then extracting from them information vital a case or to the security of the nation. While this undoubtedly happens and is an important part of computer forensics, it is most likely not the common case.

Instead the majority of forensic response comes in settings where the systems being examined are under prior control of the organization performing the investigation. There are many examples of this kind of investigation. Corporations need to respond to allegations of misuse, evidence of intrusion, or simply to preserve data in response to legal demand. Governments additionally often need to investigate potential insider activity. In these cases the computers being examined are under control of the organization, and most often the human subjects are employees and are subject to the policies of the organization.

In this paper we will present a research agenda that takes advantage of the opportunity to deploy additional software to the systems far in advance of any potential incident. This already done by some organizations which employ commercial tools on their network. These tools are typically forensic tools designed for a stand-alone investigation modified to operate in a client-server fashion over the network, such as EnCase Enterprise or Access Data Enterprise. An investigator can examine and image a system remotely, speeding the acquisition process. This saves travel time and processing time associated with physically imaging the disk. What it does not do is provide any additional information to help solve the case. Only the artifacts normally produced by the computer system are available to address hypotheses about what occurred on the system.

What is not common, and what presents a tremendous opportunity for information systems and computer scientists, is modifying systems to collect and preserve evidence *well in advance* of any investigation. We know what the most common types of investigations are. We know what evidence can be useful to prove or disprove some hypothesis. We know that digital storage is increasingly cheap, particularly compared to the time and expense involved in training or hiring forensic investigators. There seems to be no reason not to collect and store potentially relevant data ahead of time. The cost overhead is low when we can harness otherwise unused computer resources. Even when additional computing equipment is needed, the cost can be very low relative to the savings in investigative resources.

This approach raises several questions: In what situations is it possible to perform proactive preservation? What evidence should be preserved? What resources are available or required?

Computing Environment It would be absurd to expect miscreants who are misusing computers to voluntarily install ahead of time any software that would help identify and catch them. Individuals who have control over their computational resources and suspect they might be investigated should instead be expected to do the opposite and make their systems delete and overwrite evidence that would expose their actions. This is in fact the assumption that is commonly made in cases where computers are seized by investigators from outside an organization.

In the agenda being pursued in this paper, however, **we are targeting only computing environments in which the systems being investigated are not configured nor administered by the likely human subjects of the investigation**. We assume in our work that the users of the computers are employees of an organization that provides computing resources to the users. These resources are assumed to be centrally administered and as a result the software provided to the users can be selected and installed when provided. The users then have no legitimate ability to alter the system configuration. Because of this, it is reasonable to assume that proactive evidence collection can occur as configured by administrative policy without easy interference from a system user. A *proactive evidentiary collection* system could be installed that would continuously collect and preserve information.

A user might try and defeat the monitoring by taking a variety of actions. They might attempt to gain administrative access to disable any ongoing evidentiary

collection or to alter any evidence that has been collected. They might try to generate enough activity to exceed the storage resources available for proactive collection. In either case proactive evidence may be unavailable. However, we argue that in these cases we are no worse off than we would otherwise be, because even if the additional evidence was not available we would still have all of the normal operating system artifacts available. Additionally, there might be evidence of the activity that led to the compromise of the proactive collection system. This in itself could be evidence of the user's bad intentions; thus while detailed evidence would not be available, evidence of wrongdoing might be.

There also will be cases in which an outside intruder accesses a system remotely. There are two expected outcomes to this. In the first, the intruder enters the system using credentials of a non-privileged user. The intruder will thus be subject to the normal restrictions on users. In the second, the user either enters the system with administrator privileges or gains them after entry. In this the intruder will likely have the ability to modify the system software to defeat any proactive collection and potentially to alter or delete any evidence stored locally. We argue that in the first case the proactive system is a plus, as we would have evidence of the intruder's actions. In the second case, we again are no worse off than we would have been simply relying on other artifacts produced by the system.

Evidence Selection Given that we can preserve evidence in advance of an incident, what should we preserve? The answer to this question depends on the types of investigations that are expected to occur in a particular environment. If there is a specific type of incident that is expected to occur then the collection should focus on the data needed to investigate that type of incident. In the absence of a particular type of investigation we can preserve general information about user actions. This would mean preferentially preserving data created, used, or viewed by the user while limiting preservation of data that was unrelated to their activity. For example, files that were created or modified by the user would likely provide more useful information than an operating system update. Information about user commands would be more useful than details about normal system operation. At the very least, we can ensure that the normal artifacts produced by the system are retained longer than they would be otherwise.

Evidence Storage If we are to store data we clearly need to put it somewhere where a the user of the system cannot access it. We have the option of providing specific additional storage resources; reserving resources on the system; or using space for which there is no other demand. There are different costs associated with each option that need to be balanced against the potential benefit of having the data available.

Most computer systems have been shown to have significantly more storage space available than is used [8]. As the cost of disk space has decreased and the size of disks had increased this is likely more true than ever. This makes the otherwise unused space attractive for intentional retention of forensic data as the hardware cost has already been incurred. All that needs to happen is to ensure its use. There have been proposals made in the past that can be adapted to this end [7]. The downside is that the space available varies at the desire of the user. Someone who wants to limit the forensic information being kept can fill the disk, reducing the free space available.

Adding dedicated storage in the device provides a more robust but expensive option. An organization might install multiple drives into a computer, or a large drive might be split into multiple partitions. One disk or partition would be used to store forensic information and protected from the user by disallowing access using the operating system permissions. This guarantees an amount of space for forensics storage, but a user or intruder who gains administrative access can modify or delete the stored information.

The safest but most expensive storage location is on a remote server. This has several advantages. First, devices that lack sufficient internal storage, such as mobile phones, can offload the storage to where it is more convenient. Second, all forensic information is in one place making investigations simpler. The disadvantages are that the cost is higher, since server space and network connectivity are required, and the lack of a network connection can require a fallback to a local storage location.

Related Work on Proactive Evidence Collection The notion of continuous collection of data to assist in investigations is not is not new. It is common in commercial aviation, in which aircraft data and crew communications are preserved for several hours. In the event of a crash or other incident, the data is recoverable for investigation of the accident. It is increasingly common in automobiles that contain electronic that maintain information about the car and its operating conditions for extraction after a wreck.

Most operating systems also continuously collect information as part of the operating system logs. While these are valuable sources of information they are not specifically designed to support forensic investigation. As an example, the author was performing an examination of a laptop and was asked if certain files were

burned to a CD. While the logs showed when burning started and ended, it was impossible to determine what information was copied.

Other work has discussed approaches to proactive forensics, though there has been no common agreement on what the term should mean. Some have used it to mean data collection in advance of an incident to detect an insider threat based on user behavior [2, 3, 16, 19]. Others use it in the sense of continuous collection and call for the type of research we are conducting [12] and describe in the next section.

3. Preferential Data Preservation on Disk

Current forensic investigative tools and techniques gather file system and operating system artifacts to support or refute hypotheses about past user actions [5]. These tools can be very effective when artifacts related to the investigation are present. However, there is no guarantee that information relevant to some particular investigation will be preserved for any length of time.

Many forensic investigations hinge on the presence or absence of a particular file on a system. Knowing this, the subject of an investigation will often attempt to delete incriminating files. Operating system reclamation of disk blocks is somewhat random, and the longevity of such artifacts as deleted files are the result of non-deterministic system interactions. A deleted file may remain for a period of time or be quickly overwritten. Unfortunately, blocks with potentially valuable forensic information are often overwritten when there are other forensically meaningless free blocks available on the disk.

As part of this research agenda, we are working to change this by dedicating specific file system and operating system resources to maintaining forensic data that will persist over longer periods of time.

Previous Work While there has been past work in file systems to retain more complete information, none has been specifically been designed to improve forensic investigative capabilities. Instead, the most closely related work has focused on versioning and checkpointing file systems, with some work on auditable file systems. A variety of file systems have been proposed that record all changes in files over time, such as Elephant [20, 21], which was designed to allow users to undo file system actions.

On the surface, this would be be the ideal file system for forensic investigations if it were never allowed to discard old files. While Elephant does keep more information than a standard file system, Elephant was not designed for forensics and it does not keep copies of all files as its name would suggest. Instead, it keeps some files permanently, some for a limited time, and some only while they are not deleted. By design, it specifically excludes storage of things like cache files that are often a valuable investigative artifact. We believe that a file system specifically designed for forensics could provide better forensic performance at a lower system cost.

File systems that enable checkpointing [1, 6, 10, 11, 14, 15] would allow an investigator to determine if a particular file was on a file system at the time of the checkpoint; it might not be able to determine if a file was added to the system and then deleted between checkpoints. In general, we feel that the overhead of these file systems is excessive for the enterprise environment. We hypothesize that as more data is preserved, the overhead in terms of storage and processing increases proportionately.

The closest work to an forensic file system are those designed to produce verifiable audit records [17, 18]. Such a file system produces cryptographic hashes of versions of files which are sent to an auditor. Later, the auditor can verify that the files on the system were not modified based on the hashes that were committed earlier. This system shows what files have been modified or deleted prior to the audit. It does allow the auditor to determine if a particular file was present by a hash, which limits the investigation to an exact copy of the file. It does not allow similar files to be found, nor for the auditor to determine the likely contents of any particular file that is no longer present.

Cost Trade Offs From an investigative standpoint, keeping full copies of every version of every document would be an optimal approach. In an enterprise forensics environment, this approach would be prohibitively expensive. Firstly, while the cost of disks is low, the cost of administration is high. While it might be that few users would fill their disks, those that did would impose a burden. An administrator would need to refresh their machine when that occurred, resulting in lost work time for the user and a burden for the administrator who would either need to choose what to delete or update the system with a larger disk. The problems with new disks go beyond the cost of the disk; there are licensing issues with copies of operating system and software as well as the time to copy the contents between the old and new disk. Secondly, the cost of backup is already significant. For a complete forensic information record, the size of the necessary backups would increase significantly. The benefits would also be small. Most individual machines in an enterprise are never involved in an investigation. Therefore, most of the costs associated with keeping full information would never prove to be beneficial, as the

information stored will never be relevant.

We take an intermediate approach to improve the information retained for forensic investigators while not significantly increasing overall costs. While some high-security installations might choose the costs associated with full storage, we believe that our approach will be useful to the larger enterprise population.

Priority-based block recovery A common problem for forensic investigators is that use of the system degrades the availability of forensic data. This happens because a files are deleted, the disk blocks used to store that file are reclaimed for other files by the operating system. This process may not happen immediately, but the moment at which any disk block is reclaimed is not deterministic. Because of this, a recently deleted user file may be overwritten very quickly, while an unimportant temporary file may remain on the system indefinitely.

We are working to modify a file system so that blocks from user files are preserved preferentially over non-user blocks. We are creating a priority system for block recovery in which blocks from user files are left on the system as long as possible. We will test its efficacy by implementing it on the ext2 [4] file system and comparing how long a particular set of freed user data blocks are maintained for both the original and modified file systems under trace-driven simulation.

To test the usefulness and potential overhead of this approach, we will modify a Linux kernel and filesystem to implement a prototype file system that uses a two-level priority black recovery scheme. Blocks not associated with user data will be reclaimed for use before block with potential forensic data.

The ext2 filesystem [4], chosen for our experiments because it is well-understood and easy to modify, uses a number of *block groups* to store information. on the disk. Each block group contains a *superblock* that contains metadata about the block group as well as a *data block bitmap* that shows which data blocks are free or allocated within the block group. We will extend this by adding a second data block bitmap, which we call the *user data bitmap*, which will contain the location of blocks that contain data from deleted user files. We refer to these blocks as *preserved user data blocks*, or PUD blocks, and our goal is to prevent them from being overwritten as long as possible.

We will then modify the Linux kernel to support the use of this additional bitmap. Specifically, when a file is deleted, the kernel will determine who the owner of the file is. If it is a regular user, the data blocks will not be marked as available in the data block bitmap as non-user blocks will. Instead, they will be marked in the user data bitmap. When blocks are needed for a new file, the kernel will take them from the data block bitmap first and the user data bitmap only if there are no others available, preserving the user data as long as possible. Additional metadata in the superblock will track which user data is oldest.

Additionally, two fields will be added to the block group superblock: one that contains the last time that a modification was made to the user data bitmap; and a second that contains the last time that user data blocks were recovered to be overwritten.

When the kernel needs space to write a file, it will use its normal procedure to find a sufficient number of blocks on the disk, which entails examining the data block bitmap for each block group until enough unused blocks are found. In the event that there is enough free space on the disk, new files will not overwrite deleted user files. However, if a sufficient number of free blocks are not found, the OS will have to recover some of the PUD blocks pointed to by the user data bitmap. In this case, the kernel will examine the superblocks of each block group to find one which has old user data and which has not had user data recently recovered from it as indicated by the dates added to the superblock. It will then recover a large enough number of PUD blocks by removing them from the user data bit map and adding them to the data block bitmap. The algorithm by which the blocks are selected will be part of our experimentation; because the overhead associated with keeping a modification date for each individual file and the bits in the bitmap associated with it would be excessive, we hypothesize that we will be able to simulate a least-recently used approach using the two dates added to the superblock.

We believe that this approach will be effective for several reasons. First, most modern disks on individual user machines rarely approach full capacity. Because of this, it is likely that user data will be retained for a longer amount of time than under an approach where blocks were overwritten randomly. Second, users will not be able to effectively overwrite the preserved blocks because in an enterprise environment users can be prevented from having administrative access to their individual systems. However, users can mount an attack against the system by creating many large files to attempt to fill the disk to capacity. This will cause the operating system to reclaim first all available data blocks then all blocks that contain user data. While this attack will be easily detectable and would be an indication of guilt, it could eliminate other useful data. To prevent this, we will investigate using a configurable parameter that allows an administrator to specify how much of the disk should be used to keep preserved blocks. If this op-

tion is selected, the OS would not allow all such blocks to be recovered. Instead, If this option is selected the OS would keep the specified portion of the disk for PUD blocks and not allow those to be overwritten.

This approach may not be as satisfactory for systems with many users, as all user data is forced to share the same user data bitmap. This means that one active user could cause another user's preserved data blocks to be overwritten. While it would be possible to use a separate user data bitmap for each user, the overhead might be excessive and the number of users known in advance at the time the disk was formatted.

3.0.1 Performance measurement

To examine the performance of this approach, we will implement the system described above and test its performance by both benchmarking and trace testing. We will use the Andrew benchmark [13] to determine the cost of overhead compared to a stock file system.

We are currently implementing a system that can measure the effectiveness of maintaining user data by trace-driven simulation. We have low-level traces of file system access from the Harvard SOS project [9, 22]. From these traces, we recreate the state of the file system. We then seed the file system with a number of files written with recognizable patterns of data, and scan the disk to find the blocks written with our data. Next, we delete those files and subject the file system to trace-driven writing and deleting of files to simulate normal system use. Periodically we stop the traces and determine how much of the original data remains.

Our initial work will be to provide a direct comparison of existing file systems using the methodology above. We can then use that as a baseline for performance measurement.

4. Collecting Information about Deleted Files

We are also working on novel techniques to allow proactive collection of forensic information within a network. For every file that is created, copied, moved, or deleted, we create a *document signature*, which consists of metadata about the file as well as a *document fingerprint*, which is a small representation of the contents of the file. Signatures are small enough to be cheaply and securely stored, and can be queried to determine which machines on a network contain or have *ever* contained a particular file. This supports a variety of investigations, as examiners can limit expensive manual examination techniques to systems that have a high-probability of providing useful information. Using our approach, a network manager can quickly identify any system on the network that ever had a copy of a particular document, or perform a search to identify any system that ever had a documents containing a set of keywords even if the files are deleted and overwritten. The benefits of this system are lowered investigation costs and faster investigation speed, as it makes it possible to determine what set of computers needs to be examined.

The system requires that small software agents be hooked into the operating system. Figure 1 shows a diagram of the system operation, Installation of the agents occurs in parallel with training on existing documents in the network, and a dictionary of selected words, email addresses, and other tokens is created and provided to the agents. This dictionary is typically a few megabytes in size. Thereafter, as documents are deleted, modified, or copied, document signatures are created. A signature consists of information such as: the date and time of record; the file name; the user or owner information; and one or more document fingerprints, which are a small number of bits computed based on the contents of the file that form a small, non-reversible representation of the file. These signatures are stored securely, either on a remote database or locally in a manner not accessible to the user. Thereafter, searches on the document signatures can show which users or machines held particular files. Keyword searches across signatures are also possible.

Document Fingerprints and Signatures Fingerprints are at the heart of our research. They are not simple cryptographic hashes, as those are brittle and fail to match if any single bit is changed in the document. Instead, fingerprints are computed over the contents of the file in an intelligent way similar to the mechanisms that allow indexing of documents for Google and other search engines. To be effective, fingerprints must: require little storage space; be easy to compute; accurately identify files based on contents; and match files even across small edits.

To create document fingerprints, we first parse the document into individual tokens, where a token is a word, number, email address, or other series of letters and numbers. Once tokenized, tokens that are interesting or useful are kept, and others discarded. Tokens of interest may include, but are not limited to:

- All words
- Phrases
- Selective parts of speech, such as nouns or verbs
- Proper nouns: names, places, etc.

- Words longer than a fixed number of characters
- Words found within a certain set of predefined list of words
- Words based on inverse document frequencies (histogram)
- Words based on collection statistics

Given the full token set, we then sort the tokens and compare them to a dictionary of tokens developed from a training set. In practice, this set would be created from existing documents on the network during system installation. We then create a *bit vector* in which each position in the vector indicates whether a token from the dictionary was present in the file of interest or not. This bit vector is the fingerprint for that document and typically very sparse, thus compressing well. A single signature can consist of multiple types of fingerprints. For example, a signature of an email might contain a bit vector of known addresses, another of all nouns, and another of particular acronyms.

Once signatures are computed, they are stored securely where the user cannot modify or delete it. The most efficient way is to upload them a central database for ease of searching. For devices that were not online, signatures could be stored on the local file system in a location inaccessible to the user until database connectivity was available.

While our discussion has focused on files on individual computers, there are many other applications. Fingerprints can be created for emails that are sent or received, to allow tracing of email senders and receivers. A similar approach can be kept for information that is stored in any database, and signatures created when records change. Potentially, changes to virtual machine file systems could be understood and indexed as changes occur. Removable media could have signatures created as it is mounted or unmounted. Compressed or archived files could be parsed and have signatures made. It can also be possible to create and store signatures for network traffic, by using a proxy firewall and creating signatures of traffic passing through. This includes traffic like: emails entering and exiting the network, with signatures of attachments created separately; the contents of instant message conversations; the contents of file transfers; and the contents of web pages or files as they are downloaded.

Investigations Once signatures are stored, there are a variety of methods that can be used to analyze them. We expect that the most common would be to locate all instances of a particular file by creating its signature and comparing it to other known signatures, or by doing keyword searches. This will locate all devices that held a copy of that file. Note that for any file, the electronic copy is not needed and that text could be transcribed and a signature generated from the text, regardless of formatting. This ability can be very useful for a system administrator.

The direct benefit will be to simplify future forensic investigations. For a network intrusion, an examiner can recover the remnants of an attack, and then immediately determine what other machines on the network contain the same traces of attack, even if the relevant files were deleted by the attacker. Alternatively, it is possible to identify every system that ever had a particular email. For other investigations, an examiner might choose specific sets of keywords and find the machines that ever had files with those keywords. This can be helpful in determining which machines to further examine, based on a specific topic of investigation.

With the signature information centrally available, more advanced techniques are possible. We hypothesize that data mining and machine learning techniques can be used on the database of signature and user information for a wide variety of other applications. For example, user profiles of access patterns for files could be created. The profile could be done on the basis of individual patterns, or on the basis of their position or role within the organization. These would allow anomaly detection if users start accessing files outside their normal range.

We also believe that the system can be used to determine when a file logging system has failed. Many systems log accesses to restricted material, though potentially that material could be copied and distributed through mechanisms outside the logging mechanism. Our system could detect such occurrences, providing an additional layer of security for the logging system.

Current Work We are currently working on a prototype to prove the concept and allow for performance measurement. The prototype consists of four components, as shown in Figure 1:

- A mechanism for extracting text and creating a signature for a document
- An extension to the OS that can be configured to create a signature for a file when certain actions occur
- A mechanism for storing a generated signature
- A mechanism for querying a system to see if any signature matches

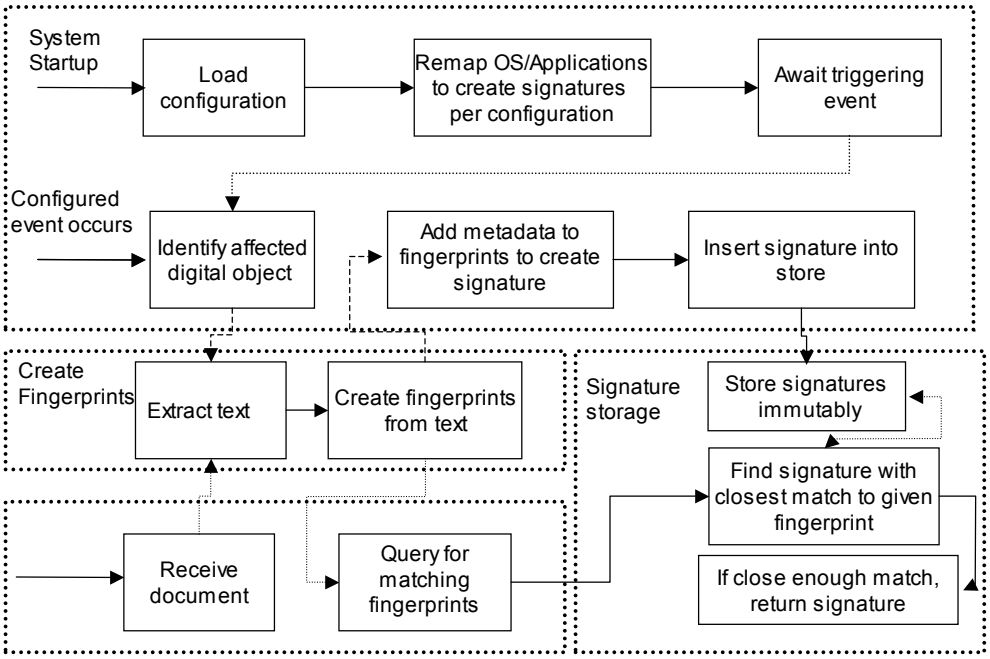

Figure 1: Signature Collection, Storage and Matching System Architecture

Our focus is on creating compact and accurate signatures The other functionality already exists commercially, or as part of operating system operation. Text extraction can be done with programs such as Oracle's Outside In technology, which can recover text from over 400 file types. Operating systems such as Windows, Linux, and Mac OS X have different mechanisms that hook into the operating system to allow modification of file operations. Storage and querying can be done with a database like MySQL or PostgreSQL.

We are therefore working to identify which fingerprints are most effective in terms of the storage space required for different types of data files. We are close to completion of an experimental test bed that allows rapid construction of different types of fingerprints and testing on a variety of data sets. We plan on developing fingerprints that are effective for emails using the Enron data set; for English text using the federal register; for foreign languages using the Trec data set; and for technical documents using a genomics data set.

Our initial results using small data sets and basic fingerprints that show we can achieve over 95% accuracy. We believe that with further investigation and experimentation we can dramatically improve this. Our prototype can create signatures for about 30 documents a second, including insertion into a PostgreSQL database. We expect that this can be greatly decreased, as we are using unoptimized java code. We can compare over 600,000 signatures a second using the same unoptimized code. Again, this gives us great confidence that we can create a fast and accurate system.

5. Conclusion

The field of computer forensics evolved in response to the needs of criminal and civil investigators. Because the field emerged to investigate existing systems most work has been on tools to better examine legacy systems, and little has been done to improve computer systems' ability to support forensic examinations, even in environments where the system is owned by the investigator.

With this paper we have shown that there exists a significant research opportunity to prepare systems to preserve information in advance. There are a variety of computing environments in which it is possible to alter the computer system ahead of any incident, including government and corporate networks where many investigations occur. To better support these settings, we am working on two distinct projects. In the first, we are examining changes to a file system to selective retain blocks from deleted files that are likely to be of interest to forensic examiners. In the second, we are applying information retrieval techniques to create small, portable signatures of files that can be used to find any system on the network that ever contained a particular document or to perform keyword searches across deleted files.

We believe that these approaches can improve the speed and accuracy of forensic investigations while lowering the overall costs. We also believe that there are a variety of other solutions that can be applied to this space, and encourage researchers in the area to examine other methods of proactive evidence collection.

References

[1] B. CORNELL, P. DINDA, F. B. Wayback: A User-level Versioning File System for Linux. In *Proceedings of the USENIX Annual Technical Conference, FREENIX Track* (2004), pp. 19–28.

[2] BRADFORD, P., BROWN, M., PERDUE, J., AND SELF, B. Towards proactive computer-system forensics. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on* (April 2004), vol. 2, pp. 648–652 Vol.2.

[3] BRADFORD, P. G., AND HU, N. A layered approach to insider threat detection and proactive forensics. In *Annual Computer Security Applications Conference (ACSAC)* (Tuscon, AZ, December 2005).

[4] CARD, R., TSO, T., AND TWEEDIE, S. Design and Implementation of the Second Extended Filesystem. In *Proceedings Dutch International Symposium on Linux* (2004).

[5] CARRIER, B. *File System Forensic Analysis*. Addison-Wesley Professional, 2005.

[6] CHUTANI, S., ANDERSON, O. T., KAZAR, M. L., AN D W. ANTHONY MASON, B. W. L., AND SIDEBOTHAM, R. N. The Episode file system. In *Proceedings of the Winter 1992 USENIX Conference* (San Francisco, CA, Winter 1992), pp. 43–60.

[7] CIPAR, J., CORNER, M. D., AND BERGER, E. D. Contributing Storage using the Transparent File System. *ACM Transactions on Storage 3*, 3 (October 2007), 12:1–12:26.

[8] DOUCEUR, J. R., AND BOLOSKY, W. J. A large-scale study of file-system contents. In *SIGMETRICS* (1999), pp. 59–70.

[9] ELLARD, D. *Trace-Based Analyses and Optimizations for Network Storage Servers*. PhD thesis, Harvard University, May 2004. Harvard Computer Science Technical Report TR-11-04.

[10] GIFFORD, D. K., NEEDHAM, R. M., AND SCHROEDER, M. D. The Cedar file system. *Communications of the ACM 31*, 3 (1988), 288–298.

[11] HAGMANN, R. Reimplementing the Cedar File System using Logging and Group Commit. In *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles* (Austin, TX, Nov. 1987), pp. 155–162. In *ACM Operating Systems Review 21*:5.

[12] HANKINS, R., AND LIU, J. Towards a forensic-aware file system. In *Electro/Information Technology, 2008. EIT 2008. IEEE International Conference on* (May 2008), pp. 84–89.

[13] HOWARD, J., KAZAR, M., MENEES, S., NICHOLS, D., SATYANARAYANAN, M., SIDEBOTHAM, R., AND WEST, M. Scale and Performance in a Distributed File System. *ACM Transactions on Computer Systems 6*, 1 (February 1988), 51–81.

[14] HOWARD, J. H., KAZAR, M. L., MENEES, S. G., NICHOLS, D. A., SATYANARAYANAN, M., SIDEBOTHAM, R. N., AND WEST, M. J. Scale and Performance in a Distributed File System. *ACM Transactions on Computer Systems 6*, 1 (Feb. 1988), 51–81.

[15] MUNISWAMY-REDDY, K., WRIGHT, C. P., HIMMER, A., AND ZADOK, E. A Versatile and User-Oriented Versioning File System. In *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST 200 4)* (San Francisco, CA, March/April 2004), pp. 115–128.

[16] PAINTSIL, A. B. Insider threat detection: A proactive forensic approach. Master's thesis, Stockholm University/The Royal Institute of Technology, Stockholm. Sweden, JMay 2007.

[17] PETERSON, Z., AND BURNS, R. Ext3cow: A Time-Shifting File System for Regulatory Compliance. *ACM Transcations on Storage 1*, 2 (2005), 190–212.

[18] PETERSON, Z. N. J., BURNS, R., AND STUBBLEFIELD, A. Limiting Liability in a Federally Compliant File System. In *Proceedings of the PORTIA Workshop on Sensitive Data in Medical, Financial, and Con tent Distribution Systems* (July 2004).

[19] RAY, D. A. *Developing a proactive digital forensics system*. PhD thesis, University of Alabama, Tuscaloosa, AL, USA, 2007.

[20] SANTRY, D. S., FEELEY, M. J., HUTCHINSON, N. C., AND VEITCH, A. C. Elephant: The File System that Never Forgets. In *Proceedings of the IEEE Workshop on Hot Topics in Operating Systems (HOTOS)* (March 1999).

[21] SANTRY, D. S., FEELEY, M. J., HUTCHINSON, N. C., VEITCH, A. C., CARTON, R. W., AND OFIR, J. Deciding When to Forget in the Elephant File System. In *Proceedings of 17th ACM Symposium on Operating Systems Principles (SOSP '99)* (December 1999).

[22] ZHU, N., CHEN, J., CHIUEH, T., AND ELLARD, D. Tbbt: Scalable and accurate trace replay for file server evaluation. In *Proceedings of ACM SIGMETRICS* (June 2005).