CERIAS Tech Report 2004-02

DEFINING AND MODELING DIGITAL EVIDENCE USING DATA FLOWS

Brian Carrier, Eugene H. Spafford

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907

Defining and Modeling Digital Evidence Using Data Flows

Brian Carrier [*] carrier@cerias.purdue.edu
Eugene H. Spafford spaf@cerias.purdue.edu

Center for Education and Research in
Information Assurance and Security - CERIAS
Purdue University
West Lafayette, IN 47907 USA

Abstract

In this paper, we define, model, and show the uses of evidence in an investigation, specifically a digital investigation. Digital evidence has been used in the courts to help prove cases, but its characteristics and role in an investigation have not been formally defined or challenged. This paper defines digital evidence by observing the role of evidence in a physical investigation, modeling the role, and applying the model to a digital investigation. The model shows the data flow between objects and how the data can be interpreted to produce information and evidence of the incident. The model can also be used to identify the source of an incident and to find additional evidence at a crime scene. The class and individual characteristics of digital evidence are given and the data flow for the 4.4 BSD kernel is used as a case study.

Key Words: Digital Evidence, Digital Investigation, Digital Forensics

1 Introduction

The role of digital evidence has been increasing in many physical and digital investigations and therefore the interest in "digital forensics" has been as well. The field of digital investigations has been largely driven by commercial products and many definitions and procedures are based on specific tools and technologies, not on general theory. Evidence plays an important role in any investigation because it can show what did or did not happen. Most forensic science disciplines have theories that are published, generally accepted, and testable, but digital forensics, or digital investigations, does not [14]. Before the research community can develop formal theories and techniques for digital investigations, a solid understanding of digital evidence must exist. This paper examines digital evidence, formally defines it, models it, and shows how the model can be used in an investigation and in future tools.

This work is based on the approach documented in [6], namely that the procedures and theories of a physical crime scene are applied to a digital crime scene. The physical world has performed investigations for thousands of years and the digital world should utilize their experience when possible. Once recognized, digital evidence is used in an investigation for the same purposes as physical evidence, but a major difference is that digital evidence is only a value and not a tangible object.

[*]Submitted to IEEE Security and Privacy 2004

To gain insight about what digital evidence actually is, we modeled evidence and applied it to digital and physical environments. The model is based on data flow between objects at a crime scene and the objects are interpreted to provide information about the incident. In a digital crime scene, the types of data flow and the types of evidence that exist are specific to the operating system and applications that are running. This paper provides the general theory of data flow in a digital device and then focuses on the 4.4 BSD kernel design.

A digital investigation is the process of preserving a digital crime scene, searching it for evidence, and reconstructing the events that lead to an incident. This has frequently been called digital forensics, but that can been seen as an abuse of terms because forensics implies science and the general process of investigating a digital device does not involve science, it involves engineering. Specific analysis techniques that are used during the digital investigation may involve science, but the general procedure is more accurately called a digital investigation [6].

A digital crime scene investigation has five major phases. The preservation phase preserves the scene by making an image of the disks or by taking measures to reduce the impact on the system. The survey phase takes a quick look at the system to find obvious pieces of evidence so that an initial hypothesis can be formed. The scene is documented and then a full search of the scene is performed to find all evidence. After the evidence from the scene has been collected, the reconstruction phase puts the pieces of evidence together so that a full theory can be developed.

In this paper, we define the characteristics of digital data and show how the evidence model can be used to find evidence during the crime scene search. The model is also used in the reconstruction phase to show the link between a suspect object and the incident being investigated. The goal of this paper is to define digital evidence and its characteristics in a simple manner that can be applied to the theory of future analysis techniques and models.

Section 2 of this paper examines evidence by first looking at its role in physical investigations and then applying that to digital investigations. The section will provide definitions for digital evidence and define its characteristics. Section 3 describes the evidence model and the details for the 4.4 BSD kernel. Section 4 uses the model to show the link between a suspect and the incident and to show how existing evidence can be used to find additional evidence. Section 5 includes a discussion of digital evidence and how it is unique.

2 Defining Evidence

2.1 Definitions

There are many existing definitions of digital evidence, but they are not general enough to be applied to a model. The International Organization on Computer Evidence (IOCE) defined digital evidence as “information stored or transmitted in binary form that may be relied upon in court [10].” The Scientific Working Group on Digital Evidence (SWGDE) uses a similar definition (and correctly removes the binary requirement) of digital evidence as “information of probative value stored or transmitted in digital form [20].” Mandia and Proisise define evidence in their incident response and investigation book as “any information of probative value, whether it confirms or dispels a matter asserted [15].”

The US Department of Justice (DOJ) defines electronic evidence as “information and data of investigative value that is stored on or transmitted by an electronic device” [22] and the UK Association of Chief Police Officers (ACPO) has a similar definition for computer-based electronic evidence as “information and data of investigative value that is stored on or transmitted by a computer [4].” In the first edition of Casey’s book, digital evidence is defined as “digital data that can establish that a crime has been committed, can provide a link between a crime and its victim, or can provide a link between a crime and its perpetrator [7].” In the second edition, the definition of digital evidence was refined to “any data stored or transmitted using a computer that supports or refutes an hypothesis of how an offense occurred or that addresses critical elements of the offense, such as intent or alibi [9].”

Any of the definitions in the preceding paragraph are acceptable for an informal discussion of digital investigations, but they are not general enough for our needs because they either focus on computers, focus on the legal system, or focus on a specific phase of an investigation. Evidence is, obviously, not unique to digital investigations. Therefore, before we make a new definition from scratch, we examine what has been defined as physical evidence.

Like digital evidence, there is no single definition of physical evidence, but many references use definitions that convey the same meaning. In *Criminalistics*, Saferstein defines physical evidence as “any and all objects that can establish that a crime has been committed or can provide a link between a crime and its victim or a crime and its perpetrator [18].” Similarly, Rynearson defines physical evidence as “any observation, relationship, or object which supports or refutes anyone’s theory of the who, how, why, what, when, and where of a crime, or which addresses the critical elements of the crime [17].” Many other references do not formally define the term.

These definitions have a common notion that evidence contains information about the crime or incident being investigated. Objects contain information about many things and Rynearson observed that “Everything is evidence of some event. The key is to identify and then capture evidence relative to the incident in question [17].”

Database and knowledge theory show us that raw objects are data. Data can be processed and interpreted to reveal information [16]. All objects at a crime scene are considered data and the investigator must interpret the objects to determine what information they can provide and determine which objects are relevant to the investigation and should be collected as evidence.

Evidence can have legal value and investigative value. The legal value of evidence is the information that can be entered into the appropriate court of law. The investigative value is the information that can be used in the process of the investigation, for example to prove or disprove a hypothesis or to find additional leads that produced relevant evidence. In general, legal requirements limit the evidence that can be used and the set of objects with legal value is a subset of the objects with investigative value. This work considers the general theory of evidence and therefore focuses on the evidence that has investigative value.

Using a combination of the previous definitions of evidence and knowledge theory, we provide a general definition of evidence. We define **evidence of an incident** as any object that contains reliable information that supports or refutes a hypothesis about the incident. The reliability and relevance of the object are evaluated during the process of interpreting the data to produce information. The definition of object can be refined to define a specific type of evidence.

Using the previous definition, we define **digital evidence of an incident** as any digital data that contain reliable information that support or refute a hypothesis about the incident. Digital evidence can be found in any digital storage medium that allows data to be written to it and the same data to be read from it. Examples of digital storage locations where digital evidence can be found include a hard disk, memory, network cables, and motherboard buses. Each location has different properties that determine how long the data resides. Currently, most evidence is collected from the hard disk and network, but monitoring devices, such as a hardware keystroke logger, may provide additional evidence. The reliability of digital evidence has been examined [8], but additional work is needed.

At some level, digital evidence has a physical form. The data on a hard disk is actually an area of a physical disk with a magnetic field, the signals in a cable are electric pulses, and wireless networks use radio waves to transmit encoded digital data. This paper focuses on the data in its raw digital form and begins at the layer where there is a stream of bits that have not been interpreted.

For completeness, we will now define other terms that are used in this paper. An **incident** is an event that violates a policy or set of laws. An **investigation of an incident** is the process of collecting relevant evidence, developing hypotheses of the incident, testing the hypotheses, and developing a final theory of the incident that is supported by the evidence. Evidence includes the objects that were not used to prove the final theory, but were used to disprove other hypotheses, called exculpatory evidence.

A **crime scene** is defined as any environment where evidence of an incident was found, even if no crime was actually committed there. For example, if a bank robber abandons his scooter on the side of the road, then that site contains evidence and is therefore a crime scene. A server that is compromised is a digital crime scene and the firewall that contains logs of the attack is also a crime scene. Note that with our definition of evidence, a person who observes an incident is evidence because she contains information about the incident (Hopefully she will not be locked up in the evidence locker until a trial).

2.2 Characteristics

Objects are used as evidence because of the information that can be derived from the interpretation of their characteristics and therefore evidence is classified by its class and its individual characteristics. Digital data can be represented in many layers of abstraction and that will impact how it is classified. For example, some cases require viewing a disk image with a hex editor and some cases are better suited by processing the file system and viewing the files and directories [5]. An investigator will decide how best to use the data and it should be classified at that level.

The **class characteristics** of an object are those that "can be associated only with a group and never with a single source [18]." Examples of class characteristics for physical objects include an object's color, size, or manufacturer. Class characteristics of digital objects include values that provide structure to the data and general values such as sizes and times. Examples of structural data include the unique identifiers of a file format (magic values) and an object's layout. Files can be classified using the type in the inode (or other meta data structure) and the extension of the file name. Table 1 shows several class characteristics for digital data in a 4.4 BSD system. The table is organized by layers of abstraction and is not comprehensive with respect to the number of abstraction layers and the characteristics within the layer.

The **individual characteristics** of an object are the characteristics that "can be associated with a common source with an extremely high degree of probability [18]." These are the characteristics that "distinguish members of the same class [12]." Examples of individual characteristics for physical objects include the markings on a bullet after it has been shot, the unique parts of a fingerprint, or the pattern from an object being torn. For digital objects, the individual characteristics are typically values that were created by a user, are not part of a file's standard structure, or are values that came from a remote system. Addresses for data structures, computers, and storage locations are also individual characteristics because they should be the only objects with that address. Table 2 shows several individual characteristics that are organized by abstraction layer.

During the five phases of an investigation, evidence goes its own phases [12]. The first phase is **recognition**, where the object is collected from the crime scene as potential evidence. This occurs during the survey and search phases of the investigation. The second phase is **identification**, where the class characteristics of the object are examined and compared to known samples to to determine the class of evidence. The final phase is **individualization**, where the individual characteristics

Table 1: Class characteristics of digital objects

Partitions and Volumes	**Network Sockets**
Type of partition	Type of protocol (UNIX, Internet, raw, TCP, UDP)
Flags in partition data structure	**Network Packets**
File Content	Length of packet
Structural contents (i.e. magic values)	Destination address or port
Allocation status of data unit	IP TTL Value
Meta Data	**Application Specific**
File type (regular, block, socket etc.)	Application type in logs
File permissions and ownership	Types of system calls in executable
File link number	Libraries required by an executable
Modified, accessed, changed, and created times	**Users**
Size of file	Groups a user is a member of
Allocation status of meta data structure	**Processes**
File Name	Parent process ID (PPID)
Parent directory name	Active terminal or console
Extension of file name	Effective and real user ID of process
Allocation status of file name structure	Start time of process

of the object are examined to determine if the object is unique among other objects in the class or to determine if the object came from the same source as another object. With computers, it is difficult to individualize digital objects to the same degree as can be done with physical objects because the digital objects are generated from instructions and there is little randomness.

3 Modeling Evidence

3.1 Overview

To show how evidence is used in an investigation, we created a graph model that showed the relationship between objects at the crime scene. We previously noted that an object is evidence if it contains information that is relevant to the investigation and that information is the result of interpreting data. Our model is a graph that uses the data flow between objects to show the relationship between them.

Let graph $G = (V, E)$ have a vertex in V for every object that has existed at the crime scene, including objects that have been removed. The directional edge (o_i, o_j) exists in E if data flowed from object o_i to object o_j. The data flow methods are different for the physical world and the digital world and will be discussed in their respective sections. Each edge is labeled with the type and time of data flow. The concept behind the edge direction is that if (o_i, o_j) exists then data flowed from o_i to o_j and o_j can be examined for evidence of o_i, although the data may no longer exist.

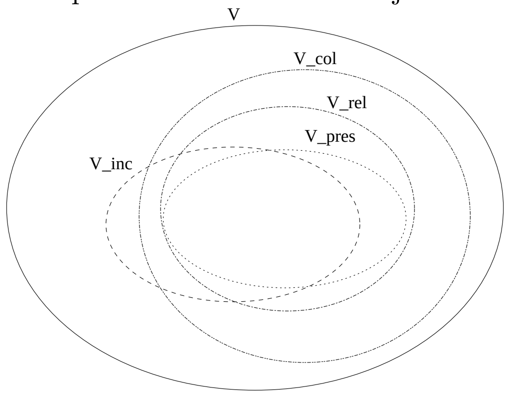
All objects can be represented at different abstraction levels. For example, a house can be one object, or there can be one object for every room, or there can be one object for every wall, floor, and ceiling. The objects in this model can be at any abstraction level, but they must be consistent. There is no inheritance in this model.

As previously noted, evidence is used in all phases of the investigation. It is collected from the scene, examined to determine if it is relevant, and then used in the reconstruction to support or

Table 2: Individual characteristics of digital objects

Partitions and Volumes	**Network Sockets**
Partition location and on-disk layout	Listening port number
File Content	IP Address of host
Address of data unit	**Network Packets**
Content not in general structure	Packet content
Hash value of all content	**Application Specific**
Meta Data	System call sequence in executables
Address of meta data structure	**Users**
File Name	User ID (UID)
Full name and path	**Processes**
	Process ID (PID)
	Memory contents not part of the generic structure

Figure 1: Graphical representation of sets of objects from the crime scene



refute hypotheses. Therefore, we defined subsets of the objects in the set V to show the investigation process. The subset $V_{inc} \subseteq V$ is the set of objects that contain information that supports the true theory of the incident. In other words, the objects in V_{inc} are those that the investigator wants to collect from the crime scene and use to prove the final theory.

The subset $V_{col} \subseteq V$ is the set of objects that the investigator collected from the crime scene as potential evidence. The objects in V_{col} are examined and used in the incident reconstruction phase where several hypotheses are considered. The subset $V_{rel} \subseteq V_{col}$ is the set of objects that are found to be relevant to the investigation and support or refute a hypothesis of the incident. The subset $V_{pres} \subseteq V_{rel}$ is the set of objects that are used in the final presentation of the incident to prove the final theory. This relationship can be seen in Figure 1.

Ideally, $V_{pres} = V_{inc}$ and the evidence that is used to support the final theory of the investigation is the same evidence that supports the true theory of the incident. If $V_{pres} \subset V_{inc}$, then there were objects at the crime scene that were not collected or there were objects that were collected from the crime scene but were incorrectly interpreted. If $V_{pres} \not\subset V_{inc}$, then there were objects that were used in the final theory that were incorrectly interpreted.

This graph can be quite large because every digital object is a vertex, but it is only a conceptual graph that shows how the objects at the crime scene are related and what objects are missing as potential evidence. The investigator never sees the entire graph and he treats it as a puzzle and tries to identify the edges between the objects that he finds at the crime scene.

3.2 Physical Evidence Model

To show how this model can be applied to the physical world, we briefly discuss how it would be used at a physical crime scene. For the physical crime scene, all physical objects are vertices. Data are observed, sometimes with the assistance of machines, using the five senses and the directional edge (o_i, o_j) is added to the graph when object o_j sees, hears, smells, tastes, or touches object o_i. All senses except touch are uni-directional because only the object that sensed the action knows that it sensed it. Touch is a special case because friction causes data to flow in both directions, so both (o_i, o_j) and (o_j, o_i) are added to the graph.

In this model, any object that picks up odors, such as fabrics, is smelling and the object can be analyzed to identify the odors. Any object that is affected by exposure to light is seeing, such as a digital camera or scanner. Note that at a molecular level, it can be argued that hearing, sight, smell, and taste are variations of touch. This model does not, by default, use that level of detail because we would then have to include every air particle in the graph. Furthermore, witnesses would have to testify in court that their ear drums vibrated because of the sound waves from the suspect's gun. If that level of abstraction is needed, then the investigator can create this graph using only touch edges and use molecules as objects.

3.3 Digital Evidence Model

3.3.1 General Theory

When the model is applied to the digital crime scene, the vertices in V are digital objects, including volatile and non-volatile digital data storage locations, processes, the kernel, and devices. The digital data are represented at the highest level of abstraction that is relevant to the incident. For example, if the incident involves writing to a specific file then the file is the object and if the incident involves writing to a sector using the raw device then the sector is the object.

Edges in the graph are created because of data flow, so we must identify data flow types in a digital environment. This paper focuses on a modern computer, but the same theory can be applied to other systems that contain digital storage. Data flow in a computer is dictated by the software and hardware, so we cannot make specific statements about all systems. In this section, we will describe the general theory and then show the data flow for the 4.4 BSD kernel and process architecture.

Evidence, and therefore this model, is concerned with stored digital data. There are few actions that can be performed on stored digital data because it is just a value. When digital data are read from memory and written to disk, no actual object is moving, just the value. Therefore, the only actions that can be performed on stored data are reading and writing. These two actions are the first two data flow types in our graph.

For reading, the edge directions are implementation specific. At a minimum, an edge (o_i, o_j) is added to the graph from the object that was read (o_i) to the object that performed the read (o_j). In some implementations, data may also flow to the object being read. For example, there could be data flow to the object being read because the disk head touches the disk platter. For the rest of this paper, we will assume that it is a uni-directional action.

When writing, the edge directions are also implementation specific. An edge (o_i, o_j) is added to the graph from the object that performed the write (o_i) to the object being written to (o_j). Some implementations may have data flow from the object being written to to the object that initiated the write, but we will assume for this paper that this is not the case.

All actions in a computer can be reduced to reading and writing data, just like all senses in the physical world can be reduced to touch. In some cases, it is more intuitive and simple to represent actions at a higher level of abstraction, especially when we are dealing with higher-level abstraction objects, such as processes and the kernel. To identify additional data flow types, we can focus on the objects that can initiate data flow. In many current operating systems, data flow can only be initiated by processes, the kernel (a special process), and hardware devices.

3.3.2 4.4 BSD Data Flow

The model was applied to the 4.4 BSD kernel because it is well understood [13] and its design is similar to many other popular operating systems. In the BSD model, processes can read and write to only their memory, processes must use system calls to request that the kernel perform operations with hardware devices, and the kernel can read and write to any memory location. The processes, kernel, and hardware devices can initiate actions on the system. We will now examine the data flow between each of these higher-level objects.

Processes can read and write to the memory locations that have been allocated to them. Processes communicate with the kernel using system calls, which will have a special edge type in the graph. Processes communicate with each other using named pipes, Unix sockets, Internet sockets, and shared memory (which map to files). System calls are used to communicate with pipes and sockets, so no new edge type is needed. A process reads from and writes to shared memory like it does to its private memory, so the basic read and write edges can be used. Processes can receive data from other processes and the kernel with a signal. As will be shown, there are other forms of data flow that are similar to signals and they will be grouped into the same edge type, an interrupt. Processes and hardware cannot directly send data to each other.

The kernel can read from and write to its own memory, the memory of any process, and devices (disks, network etc.). The kernel receives data from processes with system calls and sends data to the processes in the return value. The kernel is also responsible for receiving hardware interrupts, receiving hardware exceptions, and sending signals to processes.

Hardware devices are on the edge between the physical representation of digital data and the digital data. As previously stated, our model focuses on the lowest level of digital data and therefore we do not consider each chip and bus in the model. In this model, hardware devices, such as hard disks and network cards, can read and write data to memory using DMA, and the kernel can read and write data directly to the device. These actions will use the normal read and write edges.

Hardware devices can send data to the kernel with synchronous exceptions, which are created for an illegal instruction, divide by zero, and other cases that the currently running process needs to know about. Exceptions usually result in the kernel sending the process a signal. Hardware devices can also send data to the kernel with an asynchronous hardware interrupt, which could be created when a keyboard is typed on for example. These types of data flow are grouped with process signals and called an interrupt type.

As described, we have identified four types of data flow in the BSD design.

Read: Edge (o_i, o_j) is added to E if object o_i is read by object o_j. Object o_i is typically a data object and o_j is a process, the kernel, or a hardware device.

Write: Edge (o_i, o_j) is added to E if object o_i writes to object o_j. Object o_i is a process, the kernel, or a hardware device and o_j is typically a data object.

System Call: Edges (o_i, o_j) and (o_j, o_i) are added to E if process object o_i executes a system call to the kernel object o_j.

Interrupt: Edge (o_i, o_j) is added to E if the object o_i sends a signal, interrupt, or exception to object o_j.

The read and write actions here are presented as uni-directional. As previously mentioned, some implementations may have a bi-directional edge. Note that the `read()` and `write()` system calls may cause more than one edge to be added to the graph. For example, if a file is opened and read, then the kernel will read the data and update the access time in the file's inode. Therefore, there is a read edge from the file's content and a write edge to the inode value.

The interrupt action is used for hardware interrupts, hardware exceptions, and software signals. All of these cases have a one-way flow of information. In fact, with hardware interrupts and signals, the data is entered into a queue before the receiving object reads the data.

4 Using Evidence

In this section, we will use the evidence model to discuss two investigation techniques. For each technique, its general theory is given and then it is applied to a digital crime scene example. The first section outlines the case study, the second section describes a technique for showing the cause of an incident, and the third section describes a technique for finding evidence at a crime scene.

4.1 Case Study

This case study will be used as an example for the techniques outlined in the following sections. To simplify the example, data flow during network handshakes are ignored and individual buffers in process and kernel memory are not tracked. In this incident, an SSH server has been modified to record login and password information.

On startup, the kernel (o_1) reads the contents of the SSH server executable `/usr/sbin/sshd` (o_2) at time t_1 and creates a new process (o_3) by writing the executable contents to it at time t_2. The SSH server (o_3) opens a socket at time t_3 with the `socket()` system call and waits for a connection with the `listen()` system call at time t_4. The network card (o_4) reads data (o_5) from the network at time t_5 and sends an interrupt to the kernel (o_1) at time t_6.

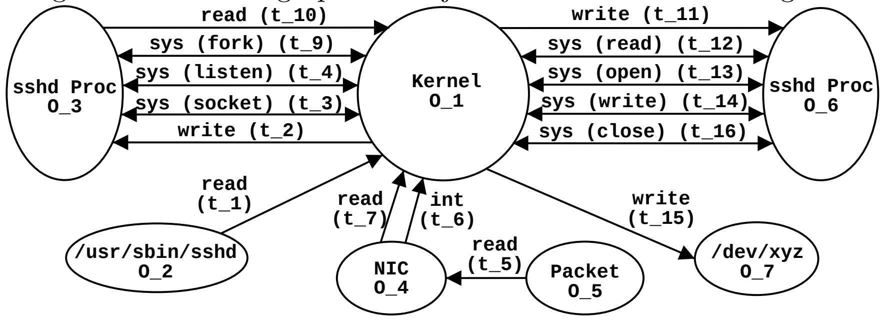
The kernel reads the data from the network device at time t_7 and removes the network headers. The packet is for the port that the SSH server opened and the kernel returns the `listen()` system call at time t_8. The SSH server uses the `fork()` system call to create a new process to handle the session at time t_9.

The kernel (o_1) creates a new child process (o_6) by reading the memory contents of the parent SSH server (o_3) at time t_{10} and writing the data to unused memory at time t_{11}. The child SSH server process issues a `read()` system call to the kernel at time t_{12} and the packet content is written to the buffer, which was passed in the `read()` system call.

An attacker modified the SSH server executable (o_2) to save login and password combinations to a log file. Therefore, the child SSH process (o_6) wrote the credentials to a log file by issuing an `open()` system call at time t_{13} for file `/dev/xyz` (o_7) and a `write()` system call at time t_{14} with the credential information. The kernel processed the system call and wrote the login and password information to the file at time t_{15}. The child process closed the file at time t_{16} with the `close()` system call.

This scenario created the graph as shown in Figure 2. Note that there would be additional edges to application files, such as the server's private SSH key, but they are left out to make the graph more simple for the example.

Figure 2: Evidence graph for trojan SSH executable investigation



4.2 Path From Source To Victim

4.2.1 Description

Evidence uses data flow to support or refute hypotheses about an incident. This section will describe how a theory that is supported by evidence can be seen in the graph because a path will exist from the object that caused the incident to the incident. We will first describe the concept and then show the technique with the digital case study.

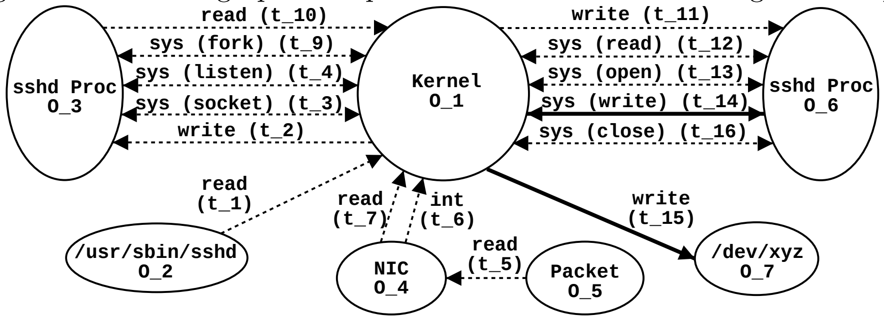
We define an **action** as an event performed by an object on one or more objects that causes data to flow to or from the other objects. Recall that we previously defined an incident as an action that is not authorized by a given policy or set of laws. Therefore, the unauthorized action has data flow edges associated with it and we can view those edges on the graph as the incident. For example, if an investigation involved the unauthorized reading of a file, then the incident would be the read edge from the file.

The **victims of an incident** are the objects that had data flow to or from them because an incident was performed on them. Data flow to the victim occurs when the incident involves an attack or modification and data flow from the victim occurs when the incident involves the unauthorized reading of a file or document. We assume that the incident being investigated caused data flow that can be represented in this model.

An action can occur by an object because it initiated the action or because an action was performed on it. For example when a process calls the `read()` system call, it causes the kernel to read the contents of a file. The process initiated the action and the kernel performed the actual read because of the action performed on it. A single action by an object can cause a chain reaction of actions, or an action chain. The **source of an incident** is the object that initiated the action chain that lead to the incident. If there is no action chain, then the object that performed the incident is the source. Note that the source of an incident is not necessarily the object that is legally guilty of a crime.

An action by object o_1 on object o_2 that causes object o_2 to perform an action on object o_3 must have data flow from object o_1 to object o_2, otherwise it would not know that an action was performed on it. Therefore, our data flow graph will show a path from the source of the incident to the object that performed the incident. A path will not exist from the source to the victim if the incident involved an action that only caused data flow from the victim (an unauthorized reading for example). Each edge in the path must have a time that is equal to or greater than the previous edge in the path. Creating the path from a suspect object to the incident edge shows that evidence exists to determine that the object caused the incident, assuming that the evidence has been found to be reliable. This technique can be used in the Reconstruction Phase of the investigation and the Presentation Phase to show the path from the suspect, if a complete path could be found.

Figure 3: Evidence graph with path from the source in the digital example



4.2.2 Case Study

In the digital example that we previously gave, the initial incident being investigated is the action that wrote the logins and passwords to the log file (o_7), edge (o_1, o_7) at time t_{15}. We have the luxury of knowing the sequence of events that led up to the incident and can therefore identify the process o_6 as the source of the incident. The kernel performed the incident action at time t_{15}, but only because of the action that was performed on it at time t_{14}. Therefore, we have the path $\langle o_6, o_1, o_7 \rangle$ from the source of the incident to the victim of the incident. The path can be seen in the graph in Figure 3. A full investigation of the incident would also examine how the process was loaded, where it was loaded from, and how the original executable was modified.

4.3 Finding Additional Evidence

4.3.1 Description

In the previous section, we saw that evidence can form a path from the source of the incident to the incident. The path was used after the evidence was found and the relevant edges and vertices in the graph were known. Unfortunately, many investigations do not have all of the nodes and edges in the graph and much of the time at the crime scene is spent searching for evidence. In this section, we will show how the characteristics of evidence can be used to find additional evidence. After the evidence has been collected and analyzed, the previous technique for identifying the object that caused the incident can be used.

The concept used in this section is not new. At a physical crime scene, investigators search for evidence using several techniques, many of which rely on geometric shapes and patterns. The link search method [12], or the logical association method [17], uses existing evidence and an investigator's experience to identify additional evidence that is related to the incident. For example, if a building was broken into and there was evidence of forced entry, then the police will search the scene for a tool that could have broken the door. If there was no evidence of forced entry, then the search focus will be placed on other objects. Of course, this technique requires an open mind so that conclusions and hypotheses are not created before enough supporting evidence has been found. The link search method allows investigators to focus their limited resources on a specific type of evidence. Fortunately, there is typically at least one piece of evidence that can be used as a starting place for the investigation; otherwise you wouldn't have an investigation.

The phenomenon of objects having information about other objects is well documented in the physical investigation world by the Locard Exchange Principle [11]. The principle states that "when two objects come into contact, a mutual exchange of matter will take place between them". This obviously does not always hold true, but is used as a motivation for examining objects for information.

Using the graph model, this search technique starts with a known piece of evidence, o_j, and examines it to identify the class and individual characteristics that it has because of data flow from other objects. Once the characteristics have been identified, the crime scene is searched for an object, o_i, that could have created the identified characteristics. If object o_i is found, then it is collected from the crime scene, added to the set V_{col}, examined to identify its unique characteristics, and the process continues until all of the existing evidence has been examined and its characteristics searched for. The process can use either a breadth-first or a depth-first search algorithm, depending on the size of the crime scene and the number of available resources. This technique can be used in the Search Phase of the investigation and can utilize the evidence that was recognized during the Survey Phase.

4.3.2 Case Study

We now apply the evidence link method to the digital example. The investigation started because an administrator noticed the regular text file in the `/dev/` directory. The file was examined and its characteristics were identified, such as its file name and some of the ASCII strings that were in the format of the file.

The investigator knew that only the kernel (o_1) could write to a file, so he examined the kernel. To search for the file name in the kernel, he used the `lsof` tool [2] to identify any file descriptors in kernel memory that had a file with the same name. No processes were found. The kernel memory was then searched for a unique ASCII string from the file format, but it was not found.

The investigator next looked for a running process that could have used a system call to cause the kernel to write to the file. The running processes were searched to find an ASCII string of the file name and the SSH server processes (o_3) and (o_6) were found. If no process was found, than any process that called the `write()` system call could have been examined. The investigator could also have identified these processes as suspect because they were one of the few processes that had access to the login and password information.

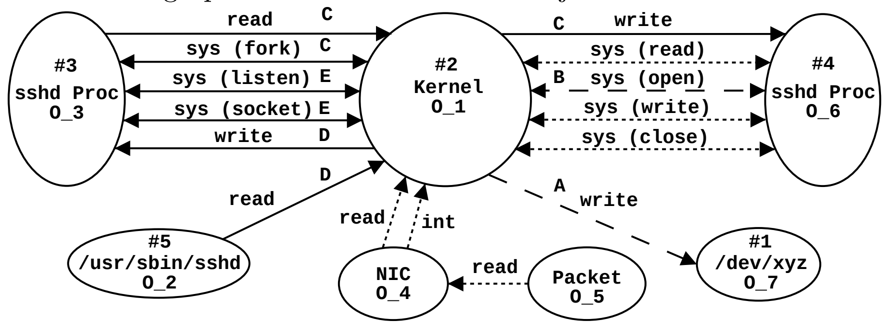
The objects for the two SSH processes were added to the set of collected objects, V_{col}, and the set of relevant objects, V_{rel}. The memory for the two processes was saved to an evidence server. The investigator knew that the kernel contained information about the process, so he used additional analysis tools to determine that o_3 was the parent process of o_6 and that the parent process (o_3) was loaded from the `/usr/sbin/sshd` (o_2) executable. The object for the executable was added to the collected objects, V_{col}, and relevant evidence, V_{rel}. The kernel also showed that the processes had TCP port 22 open and that o_3 was listening for connections.

The search for evidence continued by examining every executable on the hard disk for characteristics of the log file, but none were found. Finally, a search of all files and the unallocated space of the file system was done, but still no additional evidence was found.

The search had identified one executable and two running processes that had characteristics that could have created the log file. To identify that the executable had the ability to write passwords to that file, it was reverse engineered in a lab. The results showed that the executable could write login and password information to `/dev/xyz`. This was not a unique identifier though, because any process with the appropriate permissions could open that file and write those values, but because no other executables were found to possess the same characteristics it was determined that with a high probability it was the process that created the file.

The graph in Figure 4 shows the edges labeled with letters in the order that they were followed and the objects numbered in the order that they were found. The solid edges are those that were used to find additional evidence, the dashed edges are those that were searched for, but a link could

Figure 4: Evidence graph with search order for trojan SSH executable investigation



not be found, and the dotted edges are those that were not used in the investigation.

Note that this investigation found only the executable and process that may have been responsible for the log file, but it did not find the person that was responsible for it. That would be a followup investigation for the incident involving the modification of a system executable.

In this example, a full path was not found between the executable and the log file because the kernel no longer had information about the incident. We have information about how the process was loaded from the executable, but there is no evidence that shows that the SSH process wrote the log file. This is similar to physical investigations where a direct link is not made between the suspect and the victim. If a suspect was the only person that was seen entering the house where an attack occurred, but no evidence existed that linked the suspect to the attack, then the suspect can still be brought to court for the crime. This is sometimes called exclusive opportunity.

5 Discussion of Digital Evidence

This section discusses some of the unique aspects of digital evidence. The Introduction already discussed how digital evidence is just a value and not a tangible object that can be collected. Furthermore, passive digital evidence only exists because software developers made it exist. We define passive digital evidence as the objects that were saved by the operating system and applications and not the objects that were saved by a dedicated monitoring device. Operating systems and applications create passive evidence when the developers want it to be created and often times it is not created for the purpose of an investigation, temporary files for example. Physical evidence is generally created by the laws of nature and therefore remain consistent. A benefit of digital evidence is that additional evidence can be generated if the developer adds it to the software, but the generation of new evidence can also stop at any time if the developer removes it from the software. The notion of Locard's exchange principle in the digital world only exists when the software developers allow it to. Network sniffers and other monitoring devices can be deployed to save digital data that is not being saved by the operating system.

In addition to software developers controlling the creation of evidence, an attacker can as well. The kernel and system executables can be modified by the attacker to prevent the creation of evidence or to cause the system to create false evidence. A loadable kernel module or direct modifications to the kernel by an attacker could create characteristics in a file that would lead an investigator to a false conclusion. Modifications are not unique to the digital world, but physical world modifications can sometimes be detected by sight. It is important in all investigations to have multiple independent pieces of evidence that contain the same information.

Both physical and digital objects must be interpreted to produce information, but it is not always clear on where the boundaries of digital data are, how to interpret the data, and at what layer of abstraction. If an unknown physical object is found at a crime scene, an investigator's experience with the physical world can help him to identify what the object does, where the object begins and ends, and if it is complete. With a 100 GB hard disk and a corrupt file system, it may not be obvious where the files begin and end and what format they are. Some digital data may be able to be interpreted as two different types of data and produce valid information.

Computers have been designed to execute a set of instructions over and over again. This causes little randomness to exist in the system and therefore it is difficult to find unique characteristics to link two digital objects together, like the physical world has DNA or fingerprints. On the other hand, this makes it easier to classify data by its class characteristics.

In addition to a lack of randomness, the processes model and executable files also make the digital investigation process more difficult because there are missing links and objects. When a process starts, the kernel reads an executable file into memory, and the process performs actions on other digital objects. When that process is done, it exits and there is little remaining evidence of the process. The original executable has not changed and any evidence in memory will be lost when the system is powered off or when a new process is started. An analogy of this design in the physical world is a criminal who can create a clone of herself to commit a crime and then the clone disappears with little evidence.

This process model causes difficulties when an investigator must testify how a file was created. For example, many law enforcement investigations are based on a suspect downloading contraband files from the Internet. A defense that is being more frequently used is that a pop up add downloaded the file or that the computer was broken into and an attacker or virus put the files there [1,19]. To show that the suspect intentionally downloaded these files, an investigator may want to show that the files were created by an HTML browser and not a back door program.

If the system has been powered off since the files were downloaded, then the memory contents of the download are gone. If the downloaded file was a JPEG image, then finding unique characteristics in it that will link it to a specific HTML browser would be difficult. An exception to this is if it occurred on a Apple computer and the HFS+ file system was used because it saves the creator ID of the application that created the file [3]. In most cases though, the history files, cache, and book mark files are used to find evidence that supports the hypothesis that the suspect intentionally downloaded the files.

6 Future Work

Our future work in this area will apply this model to each of the phases in a digital investigation so that models and requirements can be developed. In our experience with conducting digital investigations and developing digital forensic tools, the model can be applied to existing technologies to make them more effective at searching for evidence. Future work will include examining each of the class and individual characteristics of digital data to identify how reliable they are. The process will use probabilities to identify the most likely source of digital data, and this moves the digital investigation process from an engineering process to a scientific process.

7 Conclusion

This work has examined and modeled digital evidence, one of the key components to a digital investigation. Digital evidence has been given many definitions and this is the first work to focus on it, model it, and formally discuss its uses. Furthermore, by viewing evidence as a result of data flow, we have shown how evidence is created and where to look for it.

The class and individual characteristics of digital data were also given so that data can be classified and the characteristics can be used to search a digital crime scene for evidence. Our experience in conducting digital investigations shows us that this evidence model applies to current technologies and can be used to develop new and more effective technologies.

The model and definition of digital evidence rely on data flow between objects. In the physical world, the data flow is dictated by the laws of nature, but in the digital world it is dictated by software and hardware. This paper documented the general theory of data flow and digital evidence and then showed the specifics for the 4.4 BSD kernel. This model can also be applied to other operating systems, such as Microsoft Windows.

Models and formal theories are useful for digital investigations because many of the same techniques that physical investigations use can be applied to digital investigations by generalizing the concepts and then applying them to the digital world. The U.S. courts rely on published and generally accepted procedures for entering scientific evidence [21], and accepted models and definitions allow important procedures to be developed, published, and accepted. This work can be used to publish and describe how existing technologies work, to develop new tools that search for related information, and to show that a digital object was the source of an incident.

References

[1] Man cleared of porn ’nightmare’. *BBC News World Edition*, Oct 2, 2003.

[2] Vic Abell. lsof v4.69. available at: http://freshmeat.net/projects/lsof/, October 16, 2003.

[3] Apple Computer Inc. *Technical Note TN1150 HFS Plus Volume Format.*

[4] Association of Chief Police Officers. *Good Practice Guide for Computer based Electronic Evidence*, 2003. available at: http://www.nhtcu.org.

[5] Brian Carrier. Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers. *International Journal of Digital Evidence*, Winter 2003.

[6] Brian Carrier and Eugene Spafford. Getting Physical With The Digital Investigation Process. *International Journal of Digital Evidence*, Fall 2003.

[7] Eoghan Casey. *Digital Evidence and Computer Crime.* Academic Press, 1 edition, 2000.

[8] Eoghan Casey. Error, Uncertainty, and Loss in Digital Evidence. *International Journal of Digital Evidence*, Summer 2002.

[9] Eoghan Casey. *Digital Evidence and Computer Crime.* Academic Press, 2 edition, 2004.

[10] International Organization on Computer Evidence. *G8 Proposed Principles For The Procedures Relating To Digital Evidence*, 2002. available at: http://www.ioce.org.

[11] Stuart James and Jon Nordby, editors. *Forensic Science: An Introduction to Scientific and Investigative Techniques.* CRC Press, 2003.

[12] Henry Lee, Timothy Palmbach, and Marilyn Miller. *Henry Lee’s Crime Scene Handbook.* Academic Press, 2001.

[13] Marshall McKusick, Keith Bostic, Michael Karels, and John Quarterman. *The Design and Implementation of the 4.4 BSD Operating System*. Addison Wesley, 1996.

[14] Gary Palmer. A Road Map for Digital Forensic Research. Technical Report DTR-T001-01, DFRWS, November 2001. Report From the First Digital Forensic Research Workshop (DFRWS).

[15] Chris Prosise and Kevin Mandia. *Incident Response: Investigating Computer Crime*. McGraw-Hill Osborne Media, 2001.

[16] Edward Quigley and Anthony Debons. Interrogative Theory of Information and Knowledge. In *Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research*, 1999.

[17] Joseph Rynearson. *Evidence and Crime Scene Reconstruction*. National Crime Investigation and Training, 6 edition, 2002.

[18] Richard Saferstein. *Criminalistics: An Introduction to Forensic Science*. Pearson, 7 edition, 2000.

[19] John Schwartz. Acquitted Man Says Virus Put Pornography on Computer. *New York Times*, Aug 11, 2003.

[20] Scientific Working Group on Digital Evidence. *ASCLD Glossary Definitions*, 2002. available at: http://www.swgde.org.

[21] Fred Smith and Rebecca Bace. *A Guide to Forensic Testimony*. Addison Wesley, 2003.

[22] Technical Working Group for Electronic Crime Scene Investigation. *Electronic Crime Scene Investigation: A Guide for First Responders*, July 2001.