Chapter 23

SYSTEM SUPPORT FOR FORENSIC INFERENCE

Ashish Gehani, Florent Kirchner and Natarajan Shankar

Abstract Digital evidence is playing an increasingly important role in prosecuting crimes. The reasons are manifold: financially lucrative targets are now connected online, systems are so complex that vulnerabilities abound and strong digital identities are being adopted, making audit trails more useful. If the discoveries of forensic analysts are to hold up to scrutiny in court, they must meet the standard for scientific evidence. Software systems are currently developed without consideration of this fact. This paper argues for the development of a formal framework for constructing "digital artifacts" that can serve as proxies for physical evidence; a system so imbued would facilitate sound digital forensic inference. A case study involving a filesystem augmentation that provides transparent support for forensic inference is described.

Keywords: Automated analysis, evidence generation, intuitionistic logic

1. Introduction

As the population density increases, the competition for resources intensifies and the probability of conflict between individuals rises commensurately. The physical and virtual worlds have markedly different mechanisms for managing the potential friction, each clearly influenced by the context in which it was developed. The security requirements of early computing systems were simple enough that they could be precisely specified and implemented [6]. This allowed criminal actions to be prevented before they occurred. For example, if a principal attempted to make an unauthorized change to a document, the reference monitor interceded and disallowed the action. In contrast, principals have greater leeway to violate rules in the physical world. However, they are retroactively held accountable for their actions. The legal system, thus, indirectly deters crime by punishing its perpetrators after they have acted.

The difference in the two approaches to handling crime is fundamental. To proactively prevent crime, it is necessary to characterize what is illegal *a priori*. Security agents must then monitor all activity and intervene when prohibited acts are in progress. An important consequence is that behavior can be policed even when users are pseudonymous. In contrast, the reactive approach relies on losses being reported by victims. The legality of an action can be adjudicated *a posteriori*. This allows complex contexts, such as intent, to be incorporated into the decision about whether specific behavior should be deemed illegal. As the population grows, the resources needed to proactively track every action of every individual become prohibitive. Reactive enforcement requires significantly less effort because the burden of monitoring is distributed among the members of the population (who are potential victims).

As computer systems grow in complexity, the limitations of the proactive model become increasingly apparent. Specifying enterprise-wide security policy is so challenging that corporations routinely outsource the task to specialized consultants [5, 13]. Characterizing attack mechanisms is an unending process, as evidenced by the need for continuous updates of intrusion detection signatures, virus bulletins and stateful firewall rules. Simultaneously, a major impediment to adopting a reactive security model is being overcome. This is the ubiquitous availability of a strong, nonrepudiable notion of identity without which accountability is meaningless. With the help of trusted platform modules [15] in commodity hardware and the mandatory use of cryptographic identities in the next-generation Internet infrastructure [4], global, certified identities will facilitate a transformation in cyber security.

In this paradigm, the creation, discovery and forensic analysis of digital evidence will play a critical role, and we must ensure that the integrity of digital evidence cannot be challenged. Consequently, it is important to create mechanisms that augment digital evidence with artifacts that are difficult to alter without detectable change. By building system support that transparently generates such metadata, digital evidence can be imbued with a level of reliability that exceeds that of evidence gathered from the physical world.

Generating forensically-sound digital evidence allows the use of a reactive security model that would yield three immediate results. First, security policy creators are relieved of the burden of defining exactly who should be allowed to do what in every possible scenario. Instead, they can define policy at a higher level of abstraction. Second, individuals have the freedom to perform a much larger range of actions in the virtual world. As in the physical world, they must be ready to justify their actions if challenged. Third, the legal semantics of particular sequences of actions can be characterized *ex post facto*. This addresses a fundamental weakness in current anomaly detection systems, which incorrectly flag activity as suspicious, especially when they are tuned to be sensitive enough to detect most real crimes. In the reactive paradigm, since no actual crime occurs, no victim will report it; this eliminates the false positives.

Once a loss is reported, the offense must be characterized. Forensic analysis in the physical world relies on the trail of environmental changes made by a crime's perpetrator. In the digital world, criminals could conceivably erase all traces of their activity. It is, therefore, incumbent upon future computing systems to provide trustworthy audit trails with sufficient detail to allow forensic conclusions to be drawn. However, the indiscriminate addition of auditing to a runtime environment introduces performance penalties for executing applications, requires large amounts of storage, and may even compromise privacy. Consequently, it is necessary to determine how to balance the needs of forensic analysis and the users of a system.

2. Standards of Evidence

The Frye legal standard [3] derives from a 1923 case where systolic blood pressure measurements were used to ascertain deception by an individual. The method was disallowed because it was not widely accepted by scientists. The Frye standard was superseded by the Federal Rules of Evidence in 1975, which require evidence to be based on scientific knowledge and to "assist the trier of fact." In 1993, the Daubert family sued Merrell Dow, claiming that its anti-nausea drug, Bendectin, caused birth defects. The case reached the U.S. Supreme Court, where the Daubert standard [19] for relevancy and reliability of scientific evidence was articulated. In order for digital evidence to be presented in court, the process used to collect it must meet the Daubert standard. In some states, the reliability of the evidence itself must be demonstrable. For example, the Texas Supreme Court extended the Daubert standard in the Havner case [14], ruling that if the "foundational data underlying opinion testimony are unreliable," then they would be considered "no evidence".

Current software systems produce *ad hoc* digital artifacts such as data file headers and audit logs. These artifacts are created by applications that may not meet the standards mentioned above, which reduces their value as evidence in legal proceedings.

We argue that software systems should be developed to automatically emit digital artifacts that cannot be forged, producing fragments whose veracity can be checked during an investigation in the same way forensic analysts currently verify physical evidence found by crime scene investigators. In addition, we suggest that a formal framework should be utilized for analyzing a collection of such digital artifacts since this will effectively codify the set of inferences that can be drawn in a court of law. The combination will allow conclusions from digital forensics to have the same weight as those drawn from physical evidence, once the reliability of digital evidence is established by precedent in court.

3. Challenges

A number of challenges must be addressed in the process of developing a framework for digital forensic analysis.

3.1 Evidence Selection

The first problem is to determine which parts of current proactive protection mechanisms can be transformed into elements in a reactive, accountability-based security apparatus. It is instructive to examine how institutions in the physical world address this issue. When the potential loss is high or the consequence is both likely and irreversible, preventive protection is often utilized. For example, a bank does not leave its vault unguarded and high-ranking public officials in the United States are provided with Secret Service protection since they are likely targets of attack.

Extant legislation already provides relevant guidelines. For example, publicly-traded companies need information flow controls to comply with the Sarbanes-Oxley Act [18], healthcare providers need data privacy protection to comply with the Health Insurance Portability and Accountability Act (HIPAA) [16], and financial services firms and educational institutions have to safeguard personal information to comply with the Gramm-Leach-Bliley Act [17].

A security system can be remodeled so that evidence of relevant activity is generated transparently. For example, instead of specifying and implementing access control for the vast majority of the data in a system, accesses and modifications can simply be recorded. The owner of a piece of data can inspect the corresponding audit trail. An owner who discerns any activity that violates the policy can initiate action against the perpetrator.

3.2 Forensic Analysis

Every crime leaves fragments of evidence. It is up to an investigator to piece the fragments together and create a hypothesis of what transpired. In doing so, the investigator must process the evidence and draw conclusions about the likelihood that the hypothesis is correct. For the operations to be considered forensically sound, at the very least they must be reproducible by the opposing counsel's experts. Consequently, a framework for analysis of the evidence must be agreed upon by all parties.

To see why it is important to have a standardized framework for reasoning about evidence, consider the effect of having different rules for operations that can be performed on digital evidence. If two operations are not commutative, then their composed output can be challenged on the grounds of the order in which they were performed. If there is agreement on commutativity, then the operations can be arbitrarily composed and the output would be considered to be acceptable. Similarly, if an operation is accepted as invertible, its input and output can be compared and checked for consistency. Any inconsistency can serve as grounds for having evidence discarded. In contrast, if an operation is not deemed to be invertible and the output is unimpeachable, then the absence of consistency between an input and output would not be grounds for eliminating the input from consideration as evidence.

Operations must be repeatable in order to meet the Daubert standard for scientific evidence. A digital forensic system must be designed to allow efficient distinctions to be made about which evidentiary properties are satisfied. For example, if a piece of evidence was derived using randomness, user input or network data, its legal admissibility will differ from the content that can be completely recomputed when persistent files are used as inputs.

3.3 Chain of Custody

When a piece of evidence is to be presented in a court, the chain of custody of the evidence must be established to guarantee that it has not been tampered with. The process makes two assumptions that do not hold by default in the virtual world. The first is that the evidence was not altered from the time it was created to the time it was collected. In a world where data is rapidly combined to produce new content, it is likely that the data found during an investigation will have already undergone editing operations before it was collected as evidence. The second assumption is that a piece of evidence was created by a single individual. A virtual object is much more likely to have multiple co-authors. Note that a co-author is a principal who owns one of the processes that operated on any of the data used to create the object in question.

In principle, these issues can be addressed by designing software functionality that transparently annotates data with the details of its provenance. If the metadata generated is imbued with nonrepudiable authenticity guarantees, it can serve as forensic evidence to establish a chain of custody. A policy verification engine can be used to infer the set of co-authors of a piece of data by inspecting a set of metadata provided as an input. It can also follow the chain of metadata attestations about modifications of the data to ensure that its evidentiary value is not negatively impacted.

4. Formal Framework

The utilization of a formal framework with an explicitly defined logic has a number of advantages over *ad hoc* analysis of digital evidence.

4.1 Standardization

The set of laws that govern forensic evidence handling and inference can be codified in the rules of the logic. The variations and precedents of each legal domain, such as a state or county, can be added as a set of augmenting axioms and rules. In particular, such standardization allows the prosecution and the defense to determine before trial what conclusions will likely be drawn by expert witnesses in court. Further, the significance of a digital artifact can be tested by checking which conclusions are dependent upon it. Thus, standardization may decrease the time to arrive at an agreement in court about which conclusions can be drawn given a body of digital evidence.

4.2 Automation

A framework that is completely defined by a formal logic can serve as a technical specification for implementing a forensic inference engine in software. In the physical world, the number of pieces of evidence introduced in court may be limited. In contrast, the number of pieces of digital evidence that may need to be utilized to draw a high-level conclusion may be significantly larger. In either case, as the number of elements that must be assembled to draw a conclusion increases, the effort to construct a sound inference grows exponentially. Automating the process will become necessary to ensure that the cost of using a body of digital evidence remains affordable in cases where the plaintiff or the defendant has a limited budget.

4.3 Soundness

Automating the process of generating nonrepudiable digital artifacts in software is likely to generate large bodies of digital evidence usable in a court of law. If the evidence must be manually assembled into a chain of inference, the likelihood of erroneous conclusions could be significant. All the pieces of evidence must be arranged into a plausible timeline, requiring numerous alternative orderings to be evaluated. Further, certain conclusions can be ruled out because the supporting evidence may be contradictory, such as being predicated on a person being in two locations at one time.

Manually verifying complex properties is likely to introduce errors that may be too subtle to identify without investing substantial resources. Automating the forensic inference process with an explicit formally defined framework guards against the introduction of such errors and ensures that the conclusions are sound.

4.4 Completeness

A formal framework that is complete yields a set of theorems that are the only possible conclusions logically inferred from the set of axioms determined by the digital evidence. If an attempt is made to draw any other conclusion, a judge can use the completeness of the forensic inference system to justify setting aside an argument on the grounds that it does not follow from the evidence.

Given a set of elements corresponding to the digital evidence and a decidable logic, an automated theorem prover can generate a sequence of all possible theorems, each corresponding to a conclusion for which a proof is available. A lawyer can examine the theorems (after filtering using suitable constraints if there are too many to inspect) to see if any of them either corroborate a hypothesis or to search for new hypotheses not previously considered. Having exhausted the set of theorems produced, the lawyer will be assured of not missing any possible line of argument using the available evidence.

5. CyberTrail

Our framework for generating and reasoning about digital artifacts is called CyberTrail. It utilizes CyberLogic [2] to reason about digital evidence. CyberLogic allows protocols to be specified as distributed logic programs that use predicates and certificates to answer trust management queries from the available evidence. Since proofs of claims are constructive, every conclusion is accompanied by an explicit chain of evidence. The logic is general enough to be utilized in a broad set of applications. In particular, its properties make it well suited for use by CyberTrail, as described below.

5.1 Digital Artifacts

A key aspect of CyberTrail is its proactive generation of digital artifacts that are as reliable as evidence gathered from the physical world. While there is no assurance that an artifact corresponding to any particular event of interest will be generated, the artifacts produced must be difficult to forge. Otherwise, the creator of an artifact could repudiate it on the grounds that someone else may have invested the effort to falsely generate the artifact.

We utilize the CyberLogic primitive for attestation to generate nonrepudiable digital artifacts. Every authority that generates attestations must have an associated signing key and verification key pair. An authority can be any entity, from a user to a piece of software. An attestation denoted by $A :\rhd S$ indicates that statement S has been signed cryptographically by authority A (using a digital signature algorithm and the signing key of authority A).

By generating digital artifacts that attest to the state and operations of a small subset of a system, CyberTrail seeks to leave a nonrepudiable trail that would allow the forensic analyst to make inferences about the global state and operations of the system. This is analogous to gathering physical evidence from a crime scene to reconstruct the events that led up to the activity that transpired during the actual crime.

5.2 Constrained Claims

Since software systems can make arbitrary claims by emitting a predicate, it is necessary to ensure that the authority to make a claim is captured in CyberTrail. This is accomplished by leveraging CyberLogic's support for bounded delegation. $\mathcal{D}_n(A_{from}, A_{to}, S)$ is generated by an agent A_{from} to indicate that A_{to} is authorized to make statement S on A_{from}'s behalf. The subscript n denotes how many successive levels A_{to} is allowed to delegate the right. By making authorization explicit, statements produced by software without accompanying evidence of delegation are constrained.

5.3 First-Order Logic

Propositional logic is not expressive enough to capture relationships where the variables must be quantified. For example, checking whether any file in the evidence was owned by a specific user u is easily expressed in first-order logic as $\exists f\ User(u) \wedge File(f) \wedge Owner(u, f)$ where the predicate $File(f)$ is true if f is a file; the predicate $User(u)$ is true if u is a user; and the predicate $Owner(u, f)$ is true if user u owns file f. CyberLogic can be extended to use higher-order logic if necessary.

5.4 Intuitionistic Logic

Classical logic, dating back to Aristotle, includes the Law of Excluded Middle, which says that a statement is either true or false (and that there is no third possibility); i.e., $S \vee \neg S$ is a tautology, where S is a statement in the logic. The original rule pertained to statements about finite sets and its was subsequently extended to infinite sets [20]. However, this gave rise to contradictions like Quine's Liar Paradox [10]. Intuitionistic logic [8] removes the Law of Excluded Middle from classical logic. The result is that the logic is better able to model the ambiguity of the real world.

Consider the statement that every user owns a file, which can be formulated as $\forall u\ \exists f\ User(u) \wedge File(f) \wedge Owner(u, f)$. Given a particular user u and a fixed set of files, it is possible to determine whether $Owner(u, f)$ holds for each f in the set. However, in general, the set of all files is not known, so it is not possible to determine whether $\exists f\ User(u) \wedge File(f) \wedge Owner(u, f)$ holds or whether $\neg(\exists f\ User(u) \wedge File(f) \wedge Owner(u, f))$ holds. If the statement $S = \exists f\ User(u) \wedge File(f) \wedge Owner(u, f)$, then in classical logic, $S \vee \neg S$ would need to hold. This property does not have to hold in intuitionistic logic, which allows us to reason about situations where the universe is open. This is important when dealing with evidence because artifacts are not limited to originating from a predefined closed universe.

5.5 Temporal Modality

A statement may hold in a limited context rather than being universally true. An authority who makes a claim about a statement may wish to convey the context explicitly. Modal logic introduces the $\Box$ and $\Diamond$ operators to indicate whether a statement is "necessarily" or "possibly" true, respectively. Temporal logic allows the validity of the statement in time to be specified. The utility is apparent when considering how to qualify the validity period of a digital certificate. CyberLogic allows an attestation to take the form $A :\!\rhd_{=t} S$ to indicate that it holds at time t. This suffices for constructing other modalities of attestation by quantifying the time. For example, $\Box A :\!\rhd S$ becomes $\forall t\ A :\!\rhd_t S$.

6. Case Study

This section describes a case study involving a filesystem augmented with CyberTrail features.

6.1 User-Space Filesystem

FUSE[12] provides a Linux kernel module that intercedes when filesystem calls are made. The call and its arguments are passed to a user-space daemon by communicating through a special character device defined for the purpose. The call can then be handled by a user-defined function. Our current prototype augments a subset of the VFS filesystem API and constructs the metadata needed for CyberTrail predicates. However, in its current form, information is inserted as records into a relational database using SQL commands. Reasoning about this digital evidence would require custom query tools built atop the SQL query interface.

Developing a new filesystem in user-space allows end users to utilize it without having to modify the current filesystem. Legacy applications can be executed and are presented with the same interface while the user-space filesystem acts as a transparent layer between the application and the native filesystem. Furthermore, errors in implementation do not crash the kernel or corrupt portions of the native filesystem that were not directly operated upon.

Since the prototype operates in user-space, it can only generate facts signed by the user (which are useful to ensure that the user does not subsequently make claims that could be repudiated by that user's earlier claims). A future extension could leverage a trusted platform module [15] to transparently construct facts without the cooperation of the user (to address a broader range of threats).

6.2 Transparent Reasoning

The next step in developing CyberTrail in the context of a filesystem is to marry the predicate generation and attestation directly with an automated reasoning environment. Every CyberLogic statement is a hereditary Harrop formula, the logical construct that is the basis for λ-Prolog [9]. We could invoke the λ-Prolog interpreter when the FUSE user-space daemon starts, and then insert predicates and attestations when the modified filesystem calls execute. By defining an inter-node communication protocol between distributed instances of the λ-Prolog interpreter, queries could be automatically resolved even when data has been modified at multiple nodes and transferred between them. In particular, subgoals could be resolved at the nodes corresponding to their targets, transparently guiding the distribution of the resolution procedure.

6.3 Granularity

If all the input data, application code and system libraries used to generate an output are available, an operation can be verified by repeating it. In practice, programs often read data from ephemeral sources such as network sockets, filesystem pipes and devices that provide sources of randomness. This prevents the output of the program from being independently validated in a distributed environment because the verifier must trust the original executor's claims about the contents derived from these sources. Since the executor has the freedom to alter the ephemeral inputs to yield a preferred output, checking the operation by repeating it does not increase the likelihood that the claimed operation was the one that was previously performed. Hence, our checks are restricted to the persistent files that are read and written by a process.

6.4 Auditing

When the system boots, an audit daemon is initialized. This maintains a table that maps process identifiers to associated metadata. The entry corresponding to each process entry contains an `accessed` list of all files that have been read by it, and a `modified` list of all files that have been written by it.

It is necessary to intercede on file `open()`, `close()`, `read()` and `write()` system calls. When a file `open()` call occurs, a check is performed to see if the calling process has an entry in the table. If not, an entry is created and populated with empty `accessed` and `modified` lists. When a `read()` operation occurs, the file being read is added to the `accessed` list of the calling process. Similarly, when a `write()` occurs, the file is added to the `modified` list of the calling process.

When a `close()` occurs, the `modified` list of the calling process is checked. If the file has actually been written to (as opposed to just being opened for writing or just having been read), the `modified` list will contain it. In this case, the `accessed` list of the process is retrieved from the table. It contains the list of files $\{i_1, \ldots, i_n\}$ that have been read during the execution of the process up to the point that the output file o was closed.

6.5 Artifact Generation

CyberTrail generates a variety of logical facts. We define a filesystem "primitive operation" to be an output file, the process that generated it and the set of input files it read in the course of its execution. For example, if a program reads a number of data sets from disk, computes a result and records it to a file, a primitive operation has been performed.

A primitive operation can be described as follows. Let o be the output file of the operation executed by user e using input files $i_1, \ldots, i_n$. If a process writes to a number of files, a separate instance of a primitive operation would represent each output file. Assume the predicates of Section 5.3 and that $Process(p)$ is true if p is a process; $Output(p, o)$ is true if file o has been written by process p; $Input(p, i)$ is true if file i has been read by process p; and $Owner(e, p)$ is true if process p has been executed by user e.

In practice, a file identifier i has the form $i = (h, f, t)$ where h is the hostname on which the file with name f was last modified at time t, in order to disambiguate files on different nodes with the same name as well as to differentiate between the state of the contents of a file at different instants of time. The identity e must have global meaning. We assume the availability of a public key infrastructure [7]. However, any distributed mechanism for resolving identities, such as linked local namespaces [1] or a web of trust [11], can be used instead.

The facts that correspond to the primitive operation are listed below. Note that CyberTrail must emit them as facts so that they can be used to resolve queries.

$$\boxed{\begin{array}{c} Process(p) \\ Owner(e, p) \\ File(o) \\ Output(p, o) \\ File(i_1) \\ Input(p, i_1) \\ \vdots \\ File(i_n) \\ Input(p, i_n) \end{array}}$$

The above step would occur after all references to the file become inactive. This possibility arises since multiple valid concurrent references may result after a single `open()` call. Such a situation occurs when a process spawns multiple threads and passes them a file descriptor. Equivalently, this occurs when a process makes a `fork()` call, creating another process that has copies of all its active file descriptors. Alternatively, this can occur if a part of the file was mapped to memory. Once all the active file descriptors are closed and the relevant memory blocks are unmapped, digital artifact generation can proceed to completion.

Digital artifacts must be difficult to forge. Therefore, the set of facts emitted above cannot serve as artifacts. They can be used during the process of forensic analysis to determine what activity had occurred in the system, but a suitable set of artifacts must also be found and assembled to validate the reconstruction. The set of digital artifacts that must be generated to accompany the set of facts listed above is given below.

$$\boxed{\begin{array}{c} e :\triangleright \mathit{Owner}(e, p) \\ e :\triangleright \mathit{Output}(p, o) \\ e :\triangleright \mathit{Input}(p, i_1) \\ \vdots \\ e :\triangleright \mathit{Input}(p, i_n) \end{array}}$$

6.6 Forensic Analysis

If CyberTrail functionality is deployed ubiquitously in software systems, the likelihood of finding digital artifacts generated by the operating system or subsequently by applications as well will increase. When a crime occurs and computer systems are involved, a forensic analyst will be able to scour the systems for digital artifacts and enter them into a database of evidence.

A forensic analyst may issue a variety of queries to the database of evidence. For example, the analyst may wish to determine the chain of custody for a piece of data. We use a query language with Prolog semantics to demonstrate the query. If i_0 denotes the file at beginning of the period of interest and i_1 denotes the same file at the end of the period of interest, the following query would verify that a complete chain of custody is available in the database:

$$\boxed{\begin{array}{l} \mathit{Chain}(i_0, i_1) := \\ \qquad \mathit{Chain}(i, i_1) \;\wedge\; \mathit{Output}(p, i) \;\wedge\; \mathit{Input}(p, i_0) \;\wedge \\ \qquad e :\triangleright \mathit{Output}(p, i) \;\wedge\; e :\triangleright \mathit{Input}(p, i_0) \end{array}}$$

On the other hand, if the analyst wishes to check if all the users who modified a particular file are known, the following query could be issued:

$$\begin{aligned}&Authors(i_0) := \\ &\quad Output(p, i_0) \;\wedge\; Owner(e, p) \;\wedge\; Input(p, i_1) \;\wedge\; Authors(i_1)\end{aligned}$$

Note that the above query does not validate the query against digital artifacts. If this is required, the query would be extended to:

$$\begin{aligned}&Authors(i_0) := \\ &\quad Output(p, i_0) \wedge Owner(e, p) \wedge Input(p, i_1) \wedge Authors(i_1) \wedge \\ &\quad e :\triangleright Output(p, i_0) \;\wedge\; e :\triangleright Owner(e, p) \;\wedge\; e :\triangleright Input(p, i_1)\end{aligned}$$

CyberTrail also enables a forensic analyst to find all the files that were derived from a particular piece of data. The following query would be issued:

$$\begin{aligned}&Derivatives(i_0) := \\ &\quad Input(p, i_0) \;\wedge\; Output(p, i_1) \;\wedge\; Derivatives(i_1)\end{aligned}$$

Similarly, a query could be constructed to check if a particular user had modified any of the data incorporated into a file. By adding more facts and artifacts, such as details of the runtime environment of a process at the time of auditing, other types of queries may be formulated. For example, a forensic analyst may wish to find all the files that were modified by an email client running under a suspect's identity during a given time period. Such a query could be constructed if the facts for the $Command(p)$ predicate were to be introduced at audit time, where $Command(p)$ is the command line used to create the process p.

7. Conclusions

Digital evidence is becoming increasingly important, but is often not sound enough to withstand court challenges. The approach described in this paper produces nonrepudiable digital artifacts in the context of filesystem operations that would enable an investigator to answer a number of forensic queries. Extensions to other software systems are complementary, enabling further inferences to be drawn from the resulting digital artifacts.

Acknowledgements

This research was partially supported by the National Science Foundation under Grant Nos. OCI-0722068 and CNS-0644783.

References

[1] M. Abadi, On SDSI's linked local name spaces, *Journal of Computer Security*, vol. 6(1-2), pp. 3–21, 1998.

[2] V. Bernat, H. Ruess and N. Shankar, First-Order CyberLogic, Technical Report, SRI International, Menlo Park, California (ftp.csl.sri .com/pub/users/shankar/cyberlogic-report.pdf), 2005.

[3] Court of Appeals of the District of Columbia, Frye v. United States, *Federal Reporter*, vol. 293, pp. 1013–1014, 1924.

[4] GENI Project Office, Global Environment for Network Innovations, BBN Technologies, Cambridge, Massachusetts (www.geni.net).

[5] International Business Machines, Security policy definition, Armonk, New York (www-935.ibm.com/services/us/index.wss/offer ing/gbs/a1002391).

[6] B. Lampson, Protection, *ACM Operating Systems Reviews*, vol. 8(1), pp. 18–24, 1974.

[7] U. Maurer, Modeling a public key infrastructure, *Proceedings of the Fourth European Symposium on Research in Computer Security*, pp. 325–350, 1996.

[8] J. Moschovakis, Intuitionistic logic, *Stanford Encyclopedia of Philosophy*, Metaphysics Research Laboratory, Stanford University, Palo Alto, California (plato.stanford.edu/entries/logic-intuitionistic).

[9] G. Nadathur, A proof procedure for the logic of hereditary Harrop formulas, *Journal of Automated Reasoning*, vol. 11(1), pp. 115–145, 1993.

[10] W. Quine, *The Ways of Paradox*, Harvard University Press, Cambridge, Massachusetts, 1962.

[11] M. Reiter and S. Stubblebine, Toward acceptable metrics of authentication, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 10–20, 1997.

[12] SourceForge, FUSE: Filesystem in userspace (fuse.sourceforge.net).

[13] Sun Microsystems, Security policy services, Santa Clara, California (www.sun.com/service/security/securitypolicyservices.xml).

[14] Supreme Court of Texas, Merrell Dow Pharmaceuticals, Inc. v. Havner, *South Western Reporter*, vol. 953(S.W.2d), pp. 706–733, 1998.

[15] Trusted Computing Group, Beaverton, Oregon (www.trustedcomp utinggroup.org).

[16] U.S. Government, Health Insurance Portability and Accountability Act, Public Law 104–191, *United States Statutes at Large*, vol. 110(3), pp. 1936–2103, 1997.

[17] U.S. Government, Gramm-Leach-Bliley Act, Public Law 106–102, 106th Congress, *United States Statutes at Large*, vol. 113(2), pp. 1338–1481, 2000.

[18] U.S. Government, Sarbanes-Oxley Act, Public Law 107–204, 107th Congress, *United States Statutes at Large*, vol. 116(1), pp. 745–810, 2003.

[19] U.S. Supreme Court, Daubert v. Merrell Dow Pharmaceuticals, Inc., *United States Reports*, vol. 509, pp. 579–601, 1983.

[20] J. van Heijenoort, *From Frege to Godel: A Source Book in Mathematical Logic 1879–1931*, Harvard University Press, Cambridge, Massachusetts, 1967.