Tracking Emigrant Data via Transient Provenance

Stephanie N. Jones Christina R. Strong Darrell D.E. Long
Ethan L. Miller
University of California, Santa Cruz

Abstract

Information leaks are a constant worry for companies and government organizations. After a leak occurs it is very important for the data owner to not only determine the extent of the leak, but who originally leaked the information. We propose a technique to extend data provenance to aid in determining potential sources of information leaks. While data provenance is commonly defined as the ancestry of a file, the ancestry recorded depends on the provenance collector. Instead of only recording where a file *came from*, we propose to also track when and where a file *leaves* the system. To track these departures, we suggest the use of *ghost objects* when a file is either written to a mounted external storage device or copied to a client machine via NFS or any other network interface such as SSH or FTP. We present our solution for tracking emigrant data and explain the minor changes to current provenance-aware storage systems required to enable our solution.

1 Introduction

Companies and government agencies have long had to protect their data from malicious users trying to destroy or modify it, or from computers losing data or modifying it. On top of that, there is an ever-present risk of data leaking out via an inside source. This data leak could be the final cut of a movie, the latest build of an operating system or a video game, all of which are examples of confidential data that would have eventually been made public. The worst type of data leak would be confidential data, or classified data in the case of a government agency, that was never intended to be released publicly. Once a data leak is discovered, the company needs to be able to identify the extent of the data leaked, and determine who caused the leak.

We assume that even the best security policies can still be compromised by a trusted party with malicious intentions. Consequently, we do not propose a way to prevent the data leak. Instead, we propose a way to use provenance to identify a set of potentially leaked files, involved users, and the time at which the leak may have occurred.

We assume that in order to leak data, a file must be copied off a central storage system prior to a public release of the data. To that end, we propose using *ghost objects*, to track when a file is moved or copied from a central storage system. These ghost objects provide a reference point for information about the file that was moved, who moved it, and when it was moved. This information is kept as the *transient provenance* for the file. Once the leaked data is public, the transient provenance information for the leaked data can be found in the provenance graph. In this way, we leverage transient provenance to track the flow of files as they emigrate from the storage system to help identify potential sources of data leaks.

We provide some background on provenance and describe the system we are using, as well as outline the assumptions we are making, in Section 2. We then discuss our solution in more detail in Section 3 and how it can be used to track emigrant data in Section 4. We outline various models for secure provenance that complement our solution in Section 5. Section 6 summarizes our solution and concludes.

2 Background

There are as many definitions of data provenance as there are techniques to collect or use it. Moreover, the collection and utilization of data provenance can be performed at multiple layers in any system. Here we discuss what provenance means within the scope of this research, and the assumptions we have made about the storage system and provenance collection.

2.1 Provenance

Despite the varied definitions of data provenance, all agree that provenance refers to the ancestry (sometimes referred to as the *lineage*) of an object, document, or file. The definition of ancestry, however, may vary greatly depending on the system in question.

For example, Hasan *et al.* define provenance to be the record of actions taken on a particular document over its lifetime and collect provenance when a file is closed after writing [4]. Each document has a provenance chain associated with it; a provenance chain consists of provenance records, created on every access of the document. The provenance record contains the identity of who accessed it, a representation of the document modifications, a cryptographic hash of the new version of the document, an integrity checksum, the user's public key, and keying material to interpret the aforementioned fields.

We refer to this type of provenance as *content-based* provenance, which records the actions taken on a document (what was changed) as well information describing those actions (who made the changes, when the changes were made). A good example of this type of provenance is the W7 model, as outlined by Ram and Liu [13]. They propose provenance is composed of the central element *what*, as well as the describing elements *who*, *why*, *when*, *where*, *how*, and *which*.

The Provenance-Aware Storage System (PASS) defines the provenance of an object to be all processes and data that influenced the final state of the object [10]. Provenance exists as a directed acyclic graph (DAG), wherein the nodes are objects containing attributes and the edges are relationships between objects. In a system like PASS, objects can be files, pipes, or processes, and only two types of relationships are recognized: *input* and *forkparent*.

We opted to use the PASS definition of provenance because it assumes that there is one central storage system that all users must communicate with, giving us a starting point from which to build on. An ideal provenance system would have content-based provenance as well, since the two types of provenance complement each other. However, this is beyond the scope of our paper and is considered to be future work. Such a system, however, would still need transient provenance to track the flow of data on and off the system.

2.2 Assumptions

We assume that files are stored on a central storage system similar to those found in the high performance computing (HPC) community. Specifically, we are assuming there is a central system on which data is both stored and manipulated, with multiple clients able to connect to it. Using provenance in an HPC environment can be helpful for the scientists that run simulations. Provenance data can be used to allow collaborating scientists to re-run experiments to verify results. Also, in an HPC environment, the input and output files can be very large. Provenance data can enable scientists to keep their input data and information workflow to recompute the results when needed. This use of provenance reduces the amount of data kept by the central storage system.

We further assume that only the central system has the ability to gather provenance information; the clients do not. We assume the clients connect to the central system using some network protocol such as NFS, CIFS, SSH, or FTP. These assumptions differ from those made by many provenance-aware systems where it is assumed that the client has provenance capabilities instead of the system, such as in the cloud [11]. We believe that our assumptions create a realistic scenario. It is far more plausible for a company or government agency to purchase a centralized storage system with provenance capabilities than for them to force their entire workforce to use a single specialized library or kernel.

As was previously asserted, we assume provenance is defined and collected in the same manner as PASS. However, PASS only keeps provenance for objects that are either stored on a PASS volume, called PASS objects, or "influence" the final state of a PASS object. Thus any file that is copied from a PASS volume will have no provenance associated with it; nor will it be recorded that the PASS file was copied off in the first place. Since PASS was originally designed for a local storage system, this was not an issue. However, in the scenario we describe, copying an object from the provenance system is a real and likely possibility. Thus, we propose using provenance to track not only where the data came from, but also if and when it emigrates beyond the control of the provenance system.

3 Transient Provenance

In order to capture information for forensic analysis in the event of a data leak, we propose the use of *ghost objects* to keep track of data that are copied or moved off the central storage system. To be more precise, a ghost object represents not a specific piece of data, but period of time during which data could be transferred off the storage system along with its probable destination. We differentiate ghost objects from regular provenance because they neither indicate a file's ancestry nor are they meant to be immutable. For these reasons that we refer to them as transient provenance.

3.1 Creating Ghosts

Our solution is geared towards two common methods of copying data off a central storage system. The first way is to copy files to an external storage device directly connected to the central storage system. The second way assumes that the user will copy data to their local machine over the network using a network protocol.

We assume that an external storage device is connected via USB, the Small Computer System Interface (SCSI), or the IEEE 1394 interface (FireWire). Each of these is a well-defined protocol that allows the host computer to recognize and mount the device that was connected.

We also intend to track all network connections to the central storage system, as any connection represents potential loss of control by the system. Should a network connection result in no files being read, then that connection can be disregarded. A network connection does not have to be initiated by a user: an application, for example, may initiate the connection on the user's behalf. This kind of connection is still tracked and is associated with the specific user, since he is the one using the data.

We assume that provenance is collected in a central storage system in the same manner as PASS; an interceptor catches system calls and sends them to an observer, which constructs provenance records and passes them to an analyzer to remove duplicates. However, as we discussed in Section 2.2, this provenance collection model will not create provenance records for files that are not stored on a PASS volume—leaving us to devise our own method for tracking illicit data copying.

When a removable storage device is connected to the central storage system or when a network connection is created, a ghost object is created to represent the connection. For our purposes, we will call the connection a *session*. The ghost object is created because the provenance system no longer has control over the data accessed during the session.

While the session is open, files which are read are added as inputs to the ghost object in the provenance graph. Each input is annotated with the timestamp of the action taken, as well as the user ID of the person who read the file. The ghost object of a removable device session is annotated with information such as the device ID and the user ID of the person who connected the device. In a network session, the ghost object is annotated with the source port, destination IP and port, process ID, and user ID.

The system considers a session to be closed when the value of an annotation changes. In a removable device session, a session is closed and a new ghost object is created when the external device is removed. In a network session, the session is closed and a new ghost object is created when the network connection is closed, a new process is started, or the user ID changes.

In this way, the system keeps track of the information flow out of the system. Each ghost object represents a period of time during which data was accessible from outside the provenance system control. The provenance graph indicates which files were accessed, who accessed them, and gives a reasonable idea of where the files went (be it a removable device or a specific IP address and port number), and information about the likely destination.

3.2 Managing Ghosts

We accept that there are scenarios where copying data off a central storage system is completely legitimate. For example, in the HPC community, a user may want to perform some post-processing on his results after his job has completed—this is commonplace and, assuming the user is authorized to access the results, completely licit. If there is a particularly popular data set, the number of ghost objects and corresponding provenance could suffer from what PASS refers to as a "provenance explosion" as data is copied off of the system.

There are several ways to mitigate this; ultimately it will need to be a system administrator who decides the correct approach for the system. One option is to specify "high importance" objects and only ghost those objects. This works well if there's only a small set of objects labeled as having "high importance" which are not easily reproducible. A good example would be the latest build of an operating system under development. It's highly unlikely that a user would be able to memorize all of the input files to recreate the build outside of the system. It would be far easier to just copy the build file itself.

Another option is to consider the ghost objects and corresponding provenance data to be transient and set up a pruning schedule. In the case of the previous operating system build example, ghost objects created in relation to the previous build version could be deleted and their provenance pruned out when the new version is created. If provenance pruning is not desired, provenance graph compression techniques [1, 15] can be used to remove duplicate information.

4 Tracking Data

By creating ghost objects when files leave the central storage system, we can use transient provenance to identify suspect users when a leak occurs. A query over the provenance graph for ghost objects related to the leaked data can help a system administrator determine a timeframe of when a file was copied from the system, a list of users who had copied the file and other data copied in the same timeframe. The list of suspect users is not

definite proof of guilt but is intended to limit the pool of suspects.

To reduce the number of false positives in the results of a query, versioning techniques can be used. As data is modified on the central storage system, its version number can be incremented. When data is found to be leaked, content-based provenance can be used to determine the leaked data's version number. The version number can be added to the ghost object query so only the users who accessed the leaked version will be returned as suspects.

4.1 Identifying Threats

Once the culprit has been identified, there are two concerns for the company: one is determining what other data could have been leaked, and the other is the possibility of the culprit having accomplices. Using a search system that incorporates provenance information, one can search for all files that the culprit, Alice, has touched. This helps identify other information that may have been leaked, since anything Alice had contact with is now potentially in the public domain.

Content-based provenance allows us to look at the changes made to each file, and identify the subset of files that were touched by Alice and other people. This could help identify potential accomplices; if Alice touches a file immediately after her coworker Bob does, or vice versa, it could indicate a relationship between Alice and Bob. If the number of files that were touched by Alice and Bob is greater than some threshold, it may be in the company's best interests to keep a close eye on what Bob does.

If a leak containing sensitive information about customers, such as a medical or credit card report, occurs, rather than determine accomplices, we wish to identify the specific people who may be impacted by the leak. This scenario is a matter of identifying the provenance of the report—all of the information that was used to create the report would be inputs, allowing the company to swiftly contact those whose information was compromised. Alternatively, if the report itself does not contain any identifying information, the transient provenance can be used to determine if any customer specific information was touched during the same session as when the report was leaked.

5 Secure Provenance

Transient provenance depends upon, and builds upon, current work to perfect secure provenance which will secure provenance data from unauthorized tampering. If the malicious user is aware of the provenance records and has access to them, there is currently nothing to stop him from modifying the provenance itself. The simple solution, then, is to not support requests that truncate or delete the provenance. If these actions are only allowed when the user taking them is physically logged into the server, then only a limited number of people have access to it—allowing it to be tracked in a different manner. There has been much discussion about the need to secure provenance; we outline several approaches we feel complement a provenance system with ghost objects.

A more advanced access control based security system is the Role-Based Access Control proposed by Chebotko *et al.* [3], with three levels of security specification. This specification is provided by a system administrator, and consists of the task level, port level, and data channel level security specifications. This allows different areas of the provenance workflow to be restricted based on access permissions. Syalim *et al.* propose [14] that the provenance graph should be stored in three relational databases; one for nodes, one for edges, and one for all possible paths. The access controls for each table can be set differently, depending on what part of the graph needs to have restricted access.

Hasan *et al.* also propose restricting access to the provenance, but they do so by having the user encrypt the changes made [4]. In this way, only users that are trusted to that user can view the changes. Their approach also prevents the current user from tampering with prior provenance records, since each provenance record contains a signature-based checksum. This is a hash of the rest of the provenance record signed with the user's private key.

Braun *et al.* discuss [2] the importance of having a security model for protecting provenance data. They outline and discuss the challenges and research questions that need to be addressed in order to provide strong security guarantees for provenance data. One such challenge is that the security level of provenance and that of the data it describes are often different. They suggest that while the attributes that describe the provenance can be secured using traditional data security models, a new type of security model is needed to secure the provenance graph itself.

The End-to-End Provenance System (EEPS) presents [8] a full scale distributed provenance architecture. In their model, provenance monitors are placed in the kernel and trusted hardware; while provenance authorities negotiate cross-domain communications. Lyle *et al.* suggest combining provenance and trusted computing together to create what they call *trusted provenance* [7]. They present a sample provenance architecture based on attestations found in trusted computing and would allow for detection of provenance that has been incorrectly recorded because of a faulty provenance collector or a malicious party.

6 Conclusion

We have proposed a novel use of provenance, in the creation of *ghost objects* to extend data provenance to include when and where a file or document leaves a central storage system. By querying these ghost objects, system administrators can identify a set of users that are potential sources of an information leak and gain a better sense for what information has been leaked.

While this work depends on the integrity of provenance data, we have shown that there are current research models that will satisfy this need. Securing provenance data is currently an open research question, and most of the focus has been on preventing modifications to provenance. With minor changes to provenance aware storage systems, we can create a storage system that will provide critical information in the event of a data leak. Not only will our ghost objects help to identify potential suspects, but they will also provide a timeframe for the leak and the set of data that might have been leaked.

Acknowledgements

This material is based upon work supported in part by: the Department of Energy under Award Number DE-FC02-10ER26017/DE-SC0005417, the Department of Energy's Petascale Data Storage Institute (PDSI) under Award Number DE-FC02-06ER25768, and the National Science Foundation under awards CCF-0937938 and IIP-0934401 (I/UCRC Center for Research in Intelligent Storage).

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] BARGA, R. S., AND DIGIAMPIETRI, L. A. Automatic capture and efficient storage of e-science experiment provenance. *Concurrency and Computation: Practice and Experience 20*, 5 (2008), 419–429.

[2] BRAUN, U., SHINNAR, A., AND SELTZER, M. Securing provenance. In *Proceedings of the 3rd conference on Hot topics in security* (Berkeley, CA, USA, 2008), USENIX Association, pp. 4:1–4:5.

[3] CHEBOTKO, A., CHANG, S., LU, S., FOTOUHI, F., AND YANG, P. Scientific workflow provenance querying with security views. In *Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management* (Washington, DC, USA, 2008), WAIM '08, IEEE Computer Society, pp. 349–356.

[4] HASAN, R., SION, R., AND WINSLETT, M. The case of the fake picasso: Preventing history forgery with secure provenance. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies* (Feb 2007).

[5] HASAN, R., SION, R., AND WINSLETT, M. Preventing history forgery with secure provenance. *Trans. Storage 5* (December 2009), 12:1–12:43.

[6] LI, T., MA, X., AND LI, N. Worm-seal: Trustworthy data retention and verification for regulatory compliance. In *Computer Security – ESORICS 2009*, M. Backes and P. Ning, Eds., vol. 5789 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2009, pp. 472–488.

[7] LYLE, J., AND MARTIN, A. Trusted computing and provenance: better together. In *Proceedings of the 2nd conference on Theory and practice of provenance* (Berkeley, CA, USA, 2010), TAPP'10, USENIX Association.

[8] MCDANIEL, P., BUTLER, K., MCLAUGHLIN, S., SION, R., ZADOK, E., AND WINSLETT, M. Towards a secure and efficient system for end-to-end provenance. In *Proceedings of the 2nd conference on Theory and practice of provenance* (Berkeley, CA, USA, 2010), TAPP'10, USENIX Association.

[9] MOREAU, L., GROTH, P., MILES, S., VAZQUEZ-SALCEDA, J., IBBOTSON, J., JIANG, S., MUNROE, S., RANA, O., SCHREIBER, A., TAN, V., AND VARGA, L. The provenance of electronic data. *Commun. ACM 51* (April 2008), 52–58.

[10] MUNISWAMY-REDDY, K.-K., HOLLAND, D. A., BRAUN, U., AND SELTZER, M. Provenance-aware storage systems. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference* (Berkeley, CA, USA, 2006), USENIX Association.

[11] MUNISWAMY-REDDY, K.-K., MACKO, P., AND SELTZER, M. Provenance for the cloud. In *Proceedings of the 8th USENIX conference on File and storage technologies* (Berkeley, CA, USA, 2010), FAST'10, USENIX Association, pp. 15–14.

[12] MUNISWAMY-REDDY, K.-K., AND SELTZER, M. Provenance as first class cloud data. *SIGOPS Oper. Syst. Rev. 43* (January 2010), 11–16.

[13] RAM, S., AND LIU, J. Understanding the semantics of data provenance to support active conceptual modeling. In *Proceedings of the Active Conceptual Modeling of Learning Workshop (ACM-L 2006) in conjunction with the 25th International Conference on Conceptual Modeling* (2006), pp. 1–12.

[14] SYALIM, A., HORI, Y., AND SAKURAI, K. Grouping provenance information to improve efficiency of access control. In *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance* (Berlin, Heidelberg, 2009), ISA '09, Springer-Verlag, pp. 51–59.

[15] XIE, Y., MUNISWAMY-REDDY, K.-K., LONG, D. D. E., AMER, A., FENG, D., AND TAN, Z. Compressing provenance graphs. In *Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance* (June 2011), TaPP'11.