From Here to Provtopia

Thomas Pasquier[1] [0000-0001-6876-1306], David Eyers[2] [0000-0002-7284-8006], and Margo Seltzer[3] [0000-0002-2165-4658]

[1] University of Bristol
[2] University of Otago
[3] University of British Columbia

Abstract. Valuable, sensitive, and regulated data flow freely through distributed systems. In such a world, how can systems plausibly comply with the regulations governing the collection, use, and management of such data? We claim that distributed data provenance, the directed acyclic graph documenting the origin and transformations of data holds the key. Provenance analysis has already been demonstrated in a wide range of applications: from intrusion detection to performance analysis. We describe how similar systems and analysis techniques are suitable both for implementing the complex policies that govern data and verifying compliance with regulatory mandates. We also highlight the challenges to be addressed to move provenance from research laboratories to production systems.

Keywords: Provenance · Distributed systems · Compliance

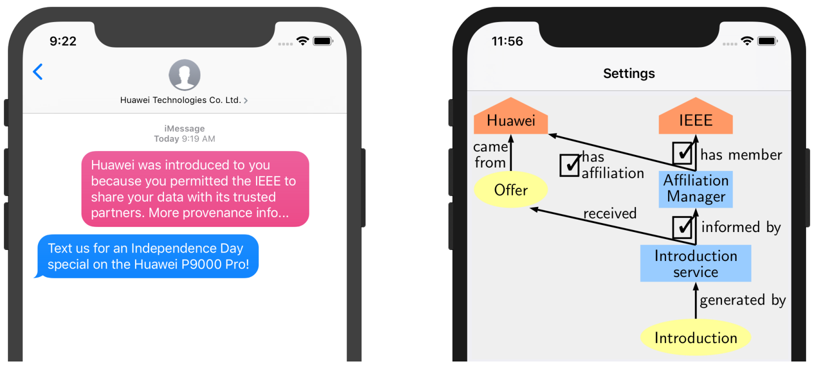
1 Vision

We live in an information economy. However, the goods on which that economy is based, i.e., the information itself, are of questionable quality. Individuals, corporations, and governments are overwhelmed with data, but the path from data to information is opaque. Imagine a different world, one we call Provtopia.

In Provtopia, the information we consume comes with a description of its composition, just as the food we buy comes with nutrition labels that alert us to allergens or additives or excessive amounts of some substance we wish to limit. Imagine that the next time you received a piece of spam, you could click on the `why` link and obtain a clear and concise explanation of why you received that email. Even better, imagine that clicking on the `never again` link ensured that you never received another piece of spam for that reason.

Now imagine that the programs and services with which we interact can also consume such labels, what will that enable? Your corporate firewall examines the labels of outgoing data and prohibits the flow of sensitive data. Or perhaps it checks to see if your customer data has been routed through a suitable aggregation mechanism, before being released to third parties. Maybe each service in your network checks the data it consumes to see if the owners of that data have authorized its use for the particular service.

Finally, imagine that we can empower users of all sorts to ask, “Where has my data been used?” whereby a graphic, such as the one shown in Fig. 1, details the flow of their information. And just as easily, they can indicate places to which they do not want their information to flow, and all future uses are prevented. We do not yet live in Provtopia, but we could.



iMessage
Today 9:19 AM

Huawei was introduced to you because you permitted the IEEE to share your data with its trusted partners. More provenance info...

Text us for an Independence Day special on the Huawei P9000 Pro!

Fig. 1. In Provtopia, the path of information is tracked and can be managed.

Provtopia exists in millions of distributed computers, large and small, that comprise our cyberinfrastructure. The currency of Provtopia is data provenance.

We begin our journey to Provtopia with a brief introduction to data provenance in the next section. We then present examples of how provenance has enabled or could enable applications ranging from security to regulatory compliance in § 3. § 4 describes the technologies that exist today and form the foundation of Provtopia. In § 5, we discuss the obstacles that lie between today’s world and Provtopia, and in § 6 we chart the path from here to there.

2 Provenance explained

Digital provenance—or just *provenance* or *lineage* or *pedigree*—is metadata detailing the origin and history of a piece of data. It is typically represented formally as relationships (interactions) among entities (data), computational activities, and the agents responsible for the generation of information or the execution of actions [11]. Its representation frequently takes the form of a directed acyclic graph (DAG) as formalised by the W3C provenance working group [6], which is derived from earlier work on the open provenance model [35]. As an illustration, Fig. 2 depicts how the W3C provenance data model represents provenance information. It depicts a scenario reporting the outcome of a continuous integration (CI) result. We have three microservices, each implemented by a different company: `git` from *gitCo*, `CI` from *SquareCI*, and `Flack` from *FlackCo*. Each service is an activity, each company is an agent, and each service is related to the company that provides it via the *associated with* relationship. The diagram shows that the Flack service is responsible for monitoring the CI process. The CI process produces reports, represented here by the *wasGeneratedBy* relationship connecting the report entity to the CI activity. The Flack activity consumes those reports, represented via the *uses* relationship connecting Flack and the report. In this case, the report came from a CI run on a particular repository, which is related to both the CI and git services: the CI service *used* the repository, which *wasGeneratedBy* git. Additionally, that instance of git was the result of a particular commit from Alice, represented by the *used* relationship between git and the commit and the *wasAttributedTo* relationship between the commit and Alice. Note that the formal models of provenance express dependencies, not information flow, so the arrows in provenance diagrams are all instances of the *depends-on* relationship.

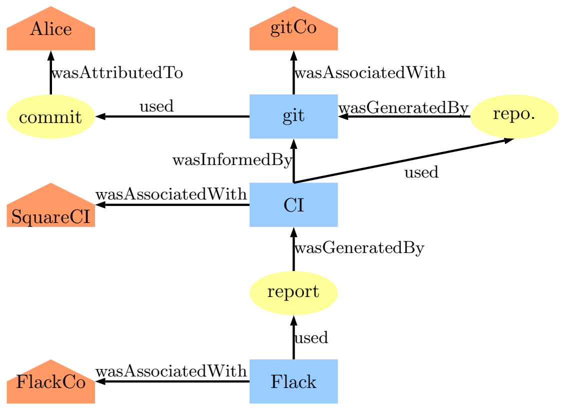


Fig. 2. A simple W3C PROV-DM provenance graph.

As a topic of research, digital provenance originated in the database community, where it provided a way to explain the results of relational database queries [8,9,12]. It later attracted attention as a means to enable reproducibility of scientific workflows, by providing a mechanism to reconstruct the computational environment from formal records of scientific computation [18,44,53,54]. Next, storage or operating system provenance emerged as a general purpose mechanism to document the transformation and flow of data within a single system [38]. Naturally, network extensions came next, and network provenance emerged as a way to debug network protocols [63]. More recently, the cybersecurity community has explored using provenance for explaining [31] and detecting system intrusions [23,24]. These applications use provenance both to explain the origin of data and to represent system execution in terms of information flow.

3 Use cases

Some of the following provenance-based applications have been implemented in prior systems, frequently using data collection infrastructures designed specifically for the application; some of have been implemented using existing provenance capture systems; and some seem possible, but have not been implemented to the best of our knowledge. Ideally, these exemplar applications provide a glimpse of what is possible in Provtopia. As a practical matter, using provenance in these applications addresses two different challenges that arise in complex distributed systems: it replaces one-off custom collection infrastructures with a single general one, and it enables an interesting class of applications for explanation, documentation, auditing, and enforcement.

3.1 Performance monitoring

Today's cloud applications are hosted on distributed systems that interact in complex ways. Performance monitoring of these systems is particularly challenging, because the root cause of an application slowdown frequently appears on a machine different from the one that reports anomalous behavior [34,59]. However, even in these complex situations, the interactions can be captured in a provenance graph. Since there must be an interaction of some sort between the faulty component and the component registering anomalous behavior, provenance can assist in being able to trace backwards along edges in the interaction graph to help pinpoint the root cause. We are aware of large cloud organizations such as eBay [57] already applying this sort of graph analysis to performance monitoring.

3.2 Debugging

Modern applications are composed of large numbers of interacting heterogeneous components (i.e., microservices). In these environments, managing the volume of diagnostic information itself becomes a challenge. Alvaro et al. [1] advocate for using provenance traces to help with such tasks, because they reveal the causal relationships buried in software system logs. Provenance data provides a consistent and systematic source of information that can expose correlated system behaviors. Automated analysis of such data has the potential to replace some of the manual debugging work that might otherwise be needed. For example, engineers debugging performance problems expend significant effort analyzing outlier events, trying to correlate them to behaviors in the software [17]. Once such outliers are detected, provenance graphs contain the necessary information to identify such correlations. While, to the best of our knowledge, no such automated tools yet exist, constructing them seems entirely feasible.

3.3 Causal Log Analysis

Provenance enables causal analysis of complex systems. Most such systems are assembled from pre-existing software components, e.g., databases and message queues, each of which may run on a different host. Such software components often already perform their own logging. However, because unexpected behavior can emerge from the interactions between such components, it is necessary to correlate data from these independent logging facilities.

Some modern provenance capture systems [42] allow information from separate software components' logs to be embedded directly into the provenance graph. The resulting provenance graph provides a qualitatively more complete view of system activity, highlighting the causal relationships between log entries that span different software components. This use case also illustrates one way that application-specific provenance (in the form of log records) can be integrated with system-level provenance to provide a complete and semantically meaningful representation of system behavior. One could imagine similar integration between multiple application-level provenance capture systems [28,39,56,61] using system provenance.

A further benefit of this approach to causal log analysis is that provenance makes information about interrelationships readily available and explicit. Traditional log analysis techniques have instead tried to infer this sort of information *post hoc*, which is potentially computationally expensive and error-prone.

3.4 Intrusion Detection

Provenance also plays a crucial role in system security. The two main types of (complementary) intrusion detection system (IDS) are (a) network-based, and (b) host-based. Existing provenance-based work focuses on host-based IDS, as even in a distributed system, an intrusion begins at one or more hosts within the network. However, the approaches described here for host-based intrusion detection can potentially be made to work in a distributed setting given existing infrastructures that can transmit provenance to an external host in real time. If many such hosts export provenance to a single analysis engine, that engine is free to implement distributed system wide detection.

It has been common practice to design host-based IDS to analyze recorded traces of system calls. Recently, though, the approach of using traces of system calls has run into difficulty correlating complex chains of events across flat logs [23]. Whole-system provenance [5,24,42] provides a source of information richer than traces of system calls, because it explicitly captures the causal relationships between events. Provenance-based approaches have shown particular promise in their ability to detect advanced persistent threats (APTs). APTs often involve multi-phase attacks, and detecting them can be difficult, due to the phases of attack often being separated by long periods of time. Solutions to APTs using system call traces have been elusive due to the challenge of correlating related events over long time periods using existing forms of log files. However, provenance can provide a compact and long-lived record that facilitates connecting the key events that are causally related. Provenance approaches that filter data at run-time can further reduce the volume of data that needs to be maintained to analyze attacks within emerging, provenance-based IDS [26,43].

3.5 Intrusion and Fault Reporting

Beyond *detecting* intrusions, or other software faults, provenance can also help visualize and explain these attacks and faults. Provenance may also help in assessing the extent of the damage caused by a leak. Such a capability may help in complying with GDPR article 33 [16], for example, which requires that users affected by breaches of personal data to be notified within a short time-window of the discovery of the breach.

Attack graphs [40,52,55], which represent computer exploits as a graph, are a common way to collect relevant information about chains of correlated activities within an attack. A provenance graph is an ideal source of data to be transformed into an attack graph to explain how an intrusion progressed and escalated.

Provenance can provide insight into the parts of the system that were affected in the process of a developing attack. The flexibility in how and what provenance data is recorded can lead to systems that capture additional context, which may help in identifying related weaknesses preemptively. These capabilities are useful in enabling system administrators to strengthen their systems, given a deeper understanding of the source of vulnerabilities.

3.6 Data Protection Compliance

The EU GDPR and similar regulations emerging in other jurisdictions place strong, enforceable protections on the use of personal data by software systems. Due to the wide-spread impact of the GDPR on existing cloud services, public attention has been drawn to both the regulation and its underlying motive—e.g., users of popular services have been notified that those services' terms and conditions have been updated both to effect the more direct simplicity and transparency required by the GDPR, and to get users' consent to use of their data.

However, there is little value in the users being given rights that they are unable to usefully exercise [41,47]. At present, most software systems that manipulate user data are largely opaque in their operations—sometimes even to experts. This makes it extremely unlikely that users will be able to fully understand where, how, when and why their data are being used.

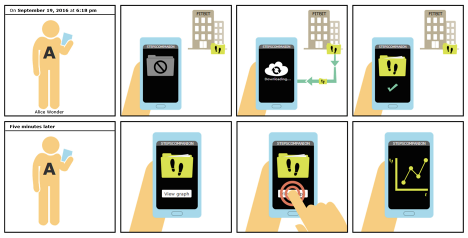


Fig. 3. Representing provenance as comic strips [51]. The comic show how Alice downloaded and visualized `fitbit` data.

Provenance can provide a useful means to explain the behavior of systems to end users, because it can provide both a high-level overview of the complete operation of a system and fine-grained details underlying each high-level operation. That said, transforming provenance into explanation, is a non-trivial task, requiring further research. Explanations from today's provenance systems are crude, presenting a user with multiple graphs and highlighting their differences or providing metrics that have no intuitive meaning to a user.

Nonetheless, promising approaches are emerging for making the behavior of distributed software systems more generally intelligible. For example, recent research demonstrated the approach of presenting data in comic-book form (see Fig. 3) [51]—not yet widely deployable, but a step in the right direction.

4 Existing Technologies

Provtopia from § 1 and the use cases from § 3 present a compelling vision of what is possible in a provenance-aware world. In this section, we outline existing technologies that can help realize this vision while § 5 identifies the areas requiring further research.

Based on our experience developing many provenance-aware applications, including some of those discussed in § 3, we propose that the key to large-scale distributed system analysis, management, and policy enforcement lies in pervasive use of whole-system provenance. First, system level provenance makes it possible to collect provenance without requiring the use of provenace-aware application and services. Second, in the presence of provenace-aware applications, system level provenance provides a mechanism to combine provenance from multiple applications in a manner that preserves causality. Third, system level provenance provides visibility into implicit interactions, for example, multiple applications that use the same data, but do not explicitly communicate. The greatest problem with system level provenance is that it does not capture application semantics. Deriving both the benefits of system level provenance and application semantics requires some form of cooperation or layering [37] among the different captures mechanisms. In § 4.2, we discuss mechanisms to facilitate such integration.

The use of whole system provenance does not require agreement on any particular provenance capture system, but does require agreement on data representation. While a provenance standard exists [6], we have found it insufficiently expressive to support applications. Instead, we have had to create our own schema on top of the standard. Nonetheless, we believe better standardization is possible, but is premature until we have significantly more widespread adoption and use of provenance. In the absence of such standardization, it is still possible today for organizations to deploy and use provenance capture systems in their own data center, without having to negotiate standards with other organizations.

We begin this section with a brief overview of whole-system provenance capture, including its practicality and how such systems can provide guarantees of completeness. Next, we discuss how to integrate provenance from different capture mechanisms, whether they be the same mechanism running on different hosts or different mechanisms on a single host, but at different layers of abstraction. Finally, we discuss the state of the art in provenance analysis.

4.1 Whole-system provenance capture

Early whole-system provenance capture systems, such as PASS [38], implemented OS-level provenance capture via system call interception and extensive, manual, and non-portable instrumentation of an existing operating system. This architecture suffered from two serious problems. First, it was unmaintainable; each new operating system release required hundreds of person hours of porting effort, and in practice, PASS died as the version of Linux on which it was based aged. Second, the fundamental architecture of system call interception is prone to security vulnerabilities relating to concurrency issues [20,58].

Hi-Fi [49] provided a better approach that relied on the reference monitor implementation in Linux: the Linux Security Module framework (LSM) [60]. Using the reference monitor—a security construct that mediates the interactions between all applications and other objects within an operating system [2]—provides increased confidence in the quality of the provenance data captured.

CamFlow built upon Hi-Fi and PASS to provide a practical and maintainable implementation of the reference monitor approach. CamFlow modularized provenance-capture, isolating it from the rest of the kernel to reduce maintenance engineering cost [42]. While PASS and Hi-Fi where one-off efforts, CamFlow has been actively maintained since 2015, has been upgraded consistently and easily with new Linux versions, and is used in several research projects. Both Hi-Fi and CamFlow use LSM, for which mediation completeness has been discussed [15,19,21,30]. Pasquier et al. elaborate on how these mediation guarantees relate to provenance completeness [43]. Further, CamFlow represents temporal relationships directly in the graph structure, rather than through metadata, improving computational performance and easing data analysis [43].

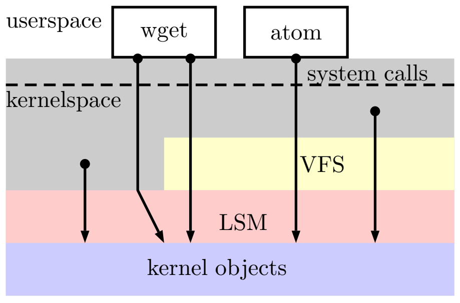


Fig. 4. Capturing provenance via the LSM framework.

Fig. 4 shows how all interactions between processes and kernel objects (e.g., files, sockets, pipes or other processes) are mediated by the LSM framework. Details of the CamFlow implementation appear in prior work [10,42,46].

4.2 Integration of Capture Mechanisms

CamFlow neatly integrates with semantically rich information from applications in two ways. First, records from existing provenance capture mechanisms such as: "big data" analytics frameworks [29], Data Science programming languages [33], network communication [62], and databases [13] can simply inject provenance records into the provenance stream [42]. Second, as discussed in § 3.3, CamFlow can incorporate application log records into its provenance stream. It connects a log event to the node representing the precise version of the thread of execution that generated it. It is then possible to correlate the surrounding graph structure to log content, enriching the system-level provenance with application-level semantics [42].

It is also possible to integrate CamFlow provenance from multiple hosts. Just as CamFlow allows applications to inject provenance records into the provenance stream, it can transmit its own system level provenance to the messaging middleware, enabling a single machine to coalesce provenance from multiple hosts. CamFlow identifies causal connections between the different hosts by analyzing the nodes representing network packets between them. This approach provides a relatively simple way to build and analyze provenance graphs that represent the execution of a distributed system [42]. We envision such a deployment within a single organization's data center. When traffic flows across multiple networks, packet labeling techniques can retain the provenance relationships (see [5]).

Summarizing the state of the art, capture mechanisms exist, they are efficient both in execution time, and storage requirements (via selective capture and storage), and system level provenance provides an easy way for them to interoperate, without requiring changes to existing infrastructure. Research-quality prototypes of all these systems exist; the only missing piece is a compelling application that convinces organizations to adopt the approach.

4.3 Provenance analysis

Early provenance capture systems assumed that provenance data must be stored and then analyzed later. This architecture introduces a significant delay between provenance capture and analysis, precluding real time applications that can prevent data movement. However, through careful design of the provenance data and graph structures, we have demonstrated that efficient runtime analysis of provenance is possible [43]. As noted in § 3, high-speed provenance analysis creates the opportunity for enforcement in addition to detection. The ability to decouple storage from analysis opens up new opportunities in the design of provenance storage as it can be removed from the critical path both of the system being monitored and the applications monitoring it.

5 From Vision to Reality

In this section we discuss areas of research into provenance systems that will support the achievement of our vision.

Machine Learning and data analytics: While provenance can be tracked within data analytics frameworks [29], understanding overall information flow is challenging when using algorithms that combine large amounts of data in relatively opaque ways. For example, it has been demonstrated that membership within a training dataset can be inferred from ML models [50]. Should this risk be presented to end-users? Are such dependencies still understandable across long chains of transformations? The interrelations between complex analytical techniques and data tracking need to be more thoroughly investigated.

Securing provenance: Provenance security is fundamental to its use in almost all applications. Provenance security encompasses confidentiality, integrity, unforgeability, non-repudiation and availability. Securing provenance is particularly difficult as the stakeholder interested in auditing a particular application may differ from those administrating or using the application. Further, provenance and the data it describes may have different security requirements [7]. For example, in a conference review scenario, the reviews (the data) should be available to authors, but the provenance of those reviews (the reviewers) should not. Conversely, we may be allowed to know the authors of a confidential report (parts of its provenance), while we may not have access to the contents of the report (the data).

Techniques exist to verify that the provenance chain has not been broken [25], and we can use hardware root of trust techniques, such as TPM [3] or vTPM [48], and remote attestation techniques [14,22] to verify the integrity of the kernel containing the provenance capture mechanism and the user space elements that manage and capture provenance data [5]. While prior work has shown that LSM mediates all flows in the kernel, there is no proof that any capture mechanism correctly and completely captures the provenance.

Storing provenance for post-hoc analysis: Recording the entire provenance graph of the execution of large distributed systems remains a significant challenge. Moyer et al. show that even a relatively modest distributed system could generate several thousand graph elements per second per machine [36]. Relatively quickly, this means handling graphs containing billions of nodes. Several options exist to reduce graph size, such as identifying and tracking only sensitive data objects [4,45] or performing property-preserving graph compression [27]. We have demonstrated that runtime provenance processing can be used to extract critical information [43], but this might not be sufficient to support some provenance needs. To our knowledge no deployment at scale has yet been attempted to demonstrate the practicality of such provenance storage approaches.

Provenance Reduction: Generating high-level execution representations (see figures 1, 2 and 3) from low-level provenance capture at the system or language level requires that provenance be trimmed and summarized. A common technique that is applied is to identify meaningful repeating patterns that can be aggregated in a single node representing a set of actions [32]. Such summarization

eases the conversion to graphical representations such as Fig. 3. Summarization may also reduce storage needs. However, potential data-loss needs to be considered carefully as important information for forensic investigations may be lost.
Provenance across administrative domains: Managing and using provenance across administrative domains introduces myriad problems, including 1) semantic interoperability, 2) transmitting (trustworthy) provenance across domains, 3) long term provenance archival and access. Although provenance standards exist [6], they are too general to support most applications. This is a vital area of investigation to build a practical deployable solution that reaches beyond a single organization.

6 Conclusion

We have introduced Provtopia—a world in which the path from data to information is annotated so as to effect its careful management and curation. In Provtopia, users have clear visibility of, and control over the use of their data by commercial organizations and government agencies. Those organizations are confident that their distributed systems comply with data handling regulation.

Despite seeming far-fetched, Provtopia is more attainable than you might expect. The provenance technologies required are emerging rapidly from many research projects, today. We provide an overview of the technologies that we and others have developed that help support this vision and highlight a number of key research challenges that remain to be addressed.

References

1. Alvaro, P., Tymon, S.: Abstracting the geniuses away from failure testing. ACM Queue **15**, 29–53 (2017)
2. Anderson, J.P.: Computer security technology planning study. Tech. Rep. ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (October 1972)
3. Bajikar, S.: Trusted Platform Module (TPM) based security on notebook PCs-white paper. Mobile Platforms Group Intel Corporation pp. 1–20 (2002)
4. Bates, A., Butler, K., Moyer, T.: Take only what you need: leveraging mandatory access control policy to reduce provenance storage costs. In: Workshop on Theory and Practice of Provenance (TaPP'15). pp. 7–7. USENIX (2015)
5. Bates, A.M., Tian, D., Butler, K.R., Moyer, T.: Trustworthy whole-system provenance for the Linux kernel. In: USENIX Security. pp. 319–334 (2015)
6. Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Moreau, L., Missier, P.e.: Prov-DM: The PROV Data Model. Tech. rep., World Wide Web Consortium (W3C) (2013), `https://www.w3.org/TR/prov-dm/`
7. Braun, U., Shinnar, A., Seltzer, M.I.: Securing provenance. In: HotSec (2008)
8. Buneman, P., Chapman, A., Cheney, J.: Provenance management in curated databases. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. pp. 539–550. ACM (2006)
9. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT. vol. 1, pp. 316–330. Springer (2001)

10. CamFlow (Accessed 22nd July 2019), `http://camflow.org/`
11. Carata, L., Akoush, S., Balakrishnan, N., Bytheway, T., Sohan, R., Selter, M., Hopper, A.: A primer on provenance. Communications of the ACM **57**(5), 52–60 (2014)
12. Cheney, J., Ahmed, A., Acar, U.A.: Provenance as dependency analysis. In: International Symposium on Database Programming Languages. pp. 138–152. Springer (2007)
13. Cheney, J., Chiticariu, L., Tan, W.C., et al.: Provenance in databases: Why, how, and where. Foundations and Trends® in Databases **1**(4), 379–474 (2009)
14. Coker, G., Guttman, J., Loscocco, P., Herzog, A., Millen, J., O'Hanlon, B., Ramsdell, J., Segall, A., Sheehy, J., Sniffen, B.: Principles of remote attestation. International Journal of Information Security **10**(2), 63–81 (2011)
15. Edwards, A., Jaeger, T., Zhang, X.: Runtime verification of authorization hook placement for the Linux security modules framework. In: Conference on Computer and Communications Security (CCS'02). pp. 225–234. ACM (2002)
16. General Data Protection Regulation. `http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC`
17. Fedorova, A., Mustard, C., Beschastnikh, I., Rubin, J., Wong, A., Miucin, S., Ye, L.: Performance comprehension at WiredTiger. In: Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 83–94. ESEC/FSE 2018, ACM (2018)
18. Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for computational tasks: A survey. Computing in Science & Engineering **10**(3) (2008)
19. Ganapathy, V., Jaeger, T., Jha, S.: Automatic placement of authorization hooks in the Linux security modules framework. In: Conference on Computer and Communications Security (CCS'05). pp. 330–339. ACM (2005)
20. Garfinkel, T., et al.: Traps and pitfalls: Practical problems in system call interposition based security tools. In: NDSS. vol. 3, pp. 163–176 (2003)
21. Georget, L., Jaume, M., Tronel, F., Piolle, G., Tong, V.V.T.: Verifying the reliability of operating system-level information flow control systems in Linux. In: International Workshop on Formal Methods in Software Engineering (FormaliSE'17). pp. 10–16. IEEE/ACM (2017)
22. Goldman, K., Perez, R., Sailer, R.: Linking remote attestation to secure tunnel endpoints. In: Workshop on Scalable Trusted Computing. pp. 21–24. ACM (2006)
23. Han, X., Pasquier, T., Seltzer, M.: Provenance-based intrusion detection: Opportunities and challenges. In: Workshop on the Theory and Practice of Provenance (TaPP 2018). USENIX (2018)
24. Han, X., Pasquier, T., Ranjan, T., Goldstein, M., Seltzer, M.: Frappuccino: Fault-detection through runtime analysis of provenance. In: Workshop on Hot Topics in Cloud Computing (HotCloud'17). USENIX (2017)
25. Hasan, R., Sion, R., Winslett, M.: The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance. In: Conference on File and Storage Technologies (FAST 09). USENIX (2009)
26. Hassan, W.U., Lemay, M., Aguse, N., Bates, A., Moyer, T.: Towards scalable cluster auditing through grammatical inference over provenance graphs. In: Network and Distributed Systems Security Symposium. Internet Society (2018)
27. Hossain, M.N., Wang, J., Sekar, R., Stoller, S.D.: Dependence-preserving data compaction for scalable forensic analysis. In: Security Symposium (USENIX Security'18). USENIX Association (2018)

28. Huang, Y., Gottardo, R.: Comparability and reproducibility of biomedical data. Briefings in Bioinformatics **14**(4), 391–401 (2012)
29. Interlandi, M., Shah, K., Tetali, S.D., Gulzar, M.A., Yoo, S., Kim, M., Millstein, T., Condie, T.: Titian: Data provenance support in Spark. Proceedings of the VLDB Endowment **9**(3), 216–227 (2015)
30. Jaeger, T., Edwards, A., Zhang, X.: Consistency analysis of authorization hook placement in the Linux security modules framework. ACM Transactions on Information and System Security (TISSEC) **7**(2), 175–205 (2004)
31. King, S.T., Chen, P.M.: Backtracking intrusions. ACM SIGOPS Operating Systems Review **37**(5), 223–236 (2003)
32. Lee, S., Niu, X., Ludäscher, B., Glavic, B.: Integrating approximate summarization with provenance capture. In: Workshop on the Theory and Practice of Provenance (TaPP 2017). USENIX (2017)
33. Lerner, B., Boose, E.: RDataTracker: collecting provenance in an interactive scripting environment. In: Workshop on the Theory and Practice of Provenance (TaPP 2014). USENIX (2014)
34. Li, J., Chen, Y., Liu, H., Lu, S., Zhang, Y., Gunawi, H.S., Gu, X., Lu, X., Li, D.: Pcatch: Automatically detecting performance cascading bugs in cloud systems. In: EuroSys '18. pp. 7:1–7:14. ACM (2018)
35. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., et al.: The open provenance model core specification (v1.1). Future generation computer systems **27**(6), 743–756 (2011)
36. Moyer, T., Gadepally, V.: High-throughput ingest of data provenance records into Accumulo. In: High Performance Extreme Computing Conference (HPEC'16). pp. 1–6. IEEE (2016)
37. Muniswamy-Reddy, K.K., Braun, U., Holland, D.A., Macko, P., MacLean, D.L., Margo, D.W., Seltzer, M.I., Smogor, R.: Layering in provenance systems. In: USENIX Annual Technical Conference (ATC'09) (2009)
38. Muniswamy-Reddy, K.K., Holland, D.A., Braun, U., Seltzer, M.I.: Provenance-aware storage systems. In: USENIX Annual Technical Conference (ATC'06). pp. 43–56 (2006)
39. Oracle Corporation: Oracle Total Recall with Oracle Database 11g release 2. `http://www.oracle.com/us/products/total-recall-whitepaper-171749.pdf` (2009)
40. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 336–345. ACM (2006)
41. Pasquier, T., Eyers, D., Bacon, J.: Personal data and the internet of things. Communications of the ACM **62**(6), 32–34 (2019)
42. Pasquier, T., Han, X., Goldstein, M., Moyer, T., Eyers, D., Seltzer, M., Bacon, J.: Practical whole-system provenance capture. In: Symposium on Cloud Computing (SoCC'17). ACM, ACM (2017)
43. Pasquier, T., Han, X., Moyer, T., Bates, A., Hermant, O., Eyers, D., Bacon, J., Seltzer, M.: Runtime analysis of whole-system provenance. In: Conference on Computer and Communications Security (CCS'18). ACM (2018)
44. Pasquier, T., Lau, M.K., Trisovic, A., Boose, E.R., Couturier, B., Crosas, M., Ellison, A.M., Gibson, V., Jones, C.R., Seltzer, M.: If these data could talk. Scientific data **4** (2017)
45. Pasquier, T., Singh, J., Bacon, J., Eyers, D.: Information flow audit for PaaS clouds. In: Cloud Engineering (IC2E), 2016 IEEE International Conference on. pp. 42–51. IEEE (2016)

46. Pasquier, T., Singh, J., Eyers, D., Bacon, J.: CamFlow: Managed data-sharing for cloud services. IEEE Transactions on Cloud Computing (2015)
47. Pasquier, T., Singh, J., Powles, J., Eyers, D., Seltzer, M., Bacon, J.: Data provenance to audit compliance with privacy policy in the internet of things. Springer Personal and Ubiquitous Computing (2018)
48. Perez, R., Sailer, R., van Doorn, L., et al.: vTPM: virtualizing the trusted platform module. In: Proc. 15th Conf. on USENIX Security Symposium. pp. 305–320 (2006)
49. Pohly, D.J., McLaughlin, S., McDaniel, P., Butler, K.: Hi-Fi: collecting high-fidelity whole-system provenance. In: Annual Computer Security Applications Conference. pp. 259–268. ACM (2012)
50. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246 (2018)
51. Schreiber, A., Struminski, R.: Tracing personal data using comics. In: International Conference on Universal Access in Human-Computer Interaction. pp. 444–455. Springer (2017)
52. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings 2002 IEEE Symposium on Security and Privacy. pp. 273–284. IEEE (2002)
53. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance techniques. Computer Science Department, Indiana University, Bloomington IN **47405**, 69 (2005)
54. Singh, G., Bharathi, S., Chervenak, A., Deelman, E., Kesselman, C., Manohar, M., Patil, S., Pearlman, L.: A metadata catalog service for data intensive applications. In: Supercomputing, 2003 ACM/IEEE Conference. pp. 33–33. IEEE (2003)
55. Swiler, L.P., Phillips, C., Ellis, D., Chakerian, S.: Computer-attack graph generation tool. In: Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01. vol. 2, pp. 307–321. IEEE (2001)
56. Tariq, D., Ali, M., Gehani, A.: Towards automated collection of application-level data provenance. In: Workshop on Theory and Practice of Provenance. pp. 16–16. TaPP'12, USENIX (2012)
57. Wang, H., Nguyen, P., Li, J., Kopru, S., Zhang, G., Katariya, S., Ben-Romdhane, S.: GRANO: Interactive graph-based root cause analysis for cloud-native distributed data platform. In: Proceedings of the 45th International Conference on Very Large Data Bases (VLDB) (2019), to appear.
58. Watson, R.N.: Exploiting concurrency vulnerabilities in system call wrappers. WOOT **7**, 1–8 (2007)
59. Whittaker, M., Alvaro, P., Teodoropol, C., Hellerstein, J.: Debugging distributed systems with why-across-time provenance. In: Symposium on Cloud Computing. pp. 333–346. SoCC '18, ACM (2018)
60. Wright, C., Cowan, C., Morris, J., Smalley, S., Kroah-Hartman, G.: Linux security module framework. In: Ottawa Linux Symposium. vol. 8032, pp. 6–16 (2002)
61. Xu, S.C., Rogers, T., Fairweather, E., Glenn, A.P., Curran, J.P., Curcin, V.: Application of data provenance in healthcare analytics software: Information visualisation of user activities. In: AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science (2018)
62. Zhou, W., Fei, Q., Narayan, A., Haeberlen, A., Loo, B.T., Sherr, M.: Secure network provenance. In: Symposium on Operating Systems Principles (SOSP'11). pp. 295–310. ACM (2011)

63. Zhou, W., Sherr, M., Tao, T., Li, X., Loo, B.T., Mao, Y.: Efficient querying and maintenance of network provenance at internet-scale. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. pp. 615–626. ACM (2010)