Model-based Abstraction of Data Provenance

Christian W. Probst
Technical University of Denmark
cwpr@dtu.dk

René Rydhof Hansen
Aalborg University
rrh@cs.aau.dk

Abstract

Identifying provenance of data provides insights to the origin of data and intermediate results, and has recently gained increased interest due to data-centric applications. In this work we extend a data-centric system view with actors handling the data and policies restricting actions. This extension is based on provenance analysis performed on system models. System models have been introduced to model and analyse spatial and organisational aspects of organisations, to identify, *e.g.*, potential insider threats. Both the models and analyses are naturally modular; models can be combined to bigger models, and the analyses adapt accordingly. Our approach extends provenance both with the origin of data, the actors and processes involved in the handling of data, and policies applied while doing so. The model and corresponding analyses are based on a formal model of spatial and organisational aspects, and static analyses of permissible actions in the models. While currently applied to organisational models, our approach can also be extended to work flows, thus targeting a more traditional model of provenance.

1. Introduction

Data-centric applications are becoming increasingly popular, promising new insights into such complex issues as, *e.g.*, consumer behaviour or stock market fluctuations. Combining data from different sources obviously might strengthen the possible results, but may pose all kinds of other problems, such as understanding the history of data and intermediate results, or being able to charge for data usage. Provenance of data provides this insight into the origin of data.

In this work we go beyond this data-centric view of a system, and extend it with a notion of actors handling the data, policies restricting access, and in general the organisation where the data is handled. This extension is based on the system model ExASyM [PH08], which has been developed to model organisations including their infrastructure, actors, processes, and access control specifications. Based on this model one can analyse organisations for potential threats posed, *e.g.*, by insiders. These analyses are primarily based on reachability in the graph representation of the system model, exploiting the fact that policies and assets are represented in the spatio-organisational model. We extend this work with annotating data with a history, which represents the actions having been performed on the data.

Our approach enables the tracing of data through organisations, extending provenance with a qualitative notion of history by adding the origin of data, the actors and processes involved in the handling of data, and policies applied while doing so. The model and analyses are based on a formal model of spatial and organisational aspects, and static analyses of permissible actions in the models. While currently applied to organisational models, our approach can also be extended to work flows, thus targeting a more traditional model of provenance. Both the models and analyses are naturally modular; models can be combined to bigger models, and the analyses adapt accordingly. This modularity supports also the combination of a system model with representations of work flows.

The rest of this paper is structured as follows. In the next section we introduce our system model and its components, followed by a discussion of the techniques for tracing provenance in Section 3. Finally, Section 4 concludes this paper and discusses some avenues for future work.

2. System Models

The ExASyM modelling formalism was initially designed with a focus on modelling not only the logical structure of an organisation and the actors interacting with or within the organisation, but also the *physical* infrastructure, enabling convenient and precise modelling of organisations and their physical environments. ExASyM models consist of separate domains for infrastructure (physical objects), actors (processes and humans), and data items (data and keys) [PH08, PHN07]. Elements that belong to different domains can be connected: actors can be at different locations or move between them; data can be carried by actors or be placed in a location. A number of analyses can be applied to ExASyM models to reason about different security properties [PH08], including graph-based analyses and fully automated static program analyses [PHN07]. The semantics of ExASyM is formalised using a variant of the Klaim family of process calculi [BLP02, dNFP98, GP03]. In addition to facilitating formal reasoning and proofs of correctness, the formal semantics also provides a foundation for developing advanced analyses and the application of formal methods. Other system models with similar goals and formal foundations include Portunes [DPH10] and ANKH [Pie11].

Due to space reasons we omit the definition of syntax and most of the semantics of our system model. The important aspects for data provenance analysis are that nodes can have different domains such as physical and virtual, and that processes can cross domain boundaries, *e.g.*, to start a process, and that policies consist of pairs of required credentials and enabled actions. The credentials can be the process' identity, data possessed by the process, or its location, the actions can be input, output, execute, or move. Figure 1 shows exemplarily the semantics of the actions in acKlaim. Before an action may be performed by the semantics, a reference monitor checks whether the action from is a neighbouring location and that the actor who wants to perform the action has the necessary rights to discharge the policy.

While the reference monitor in the semantics for a given actor checks whether a certain action is permissible, our approach is reverse — we assume that the actors who can perform an action might do so. The analysis only needs to identify who could have executed an action.

The exemplary system model shown in Figure 2 represents a small organisation with an office, a server room, and a computer network. The computer *PC2* in the server room is connected to the Internet. Access to the organisation is based on policies checking

$$\frac{[\![t]\!] = et \qquad \boxed{\langle \mathcal{I}, n, \kappa\rangle \rightsquigarrow (l, l', \mathbf{o})}}{l ::^{\delta} [\mathbf{out}\,(t)\,@l'.P]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} [P']^{\langle n',\kappa'\rangle} \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} [P']^{\langle n',\kappa'\rangle} \parallel l' ::^{\delta'} \langle et\rangle}$$

$$\frac{\mathrm{match}([\![T]\!], et) = \sigma \qquad \boxed{\langle \mathcal{I}, n, \kappa\rangle \rightsquigarrow (l, l', \mathbf{i})}}{l ::^{\delta} [\mathbf{in}\,(T)\,@l'.P]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} \langle et\rangle \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P\sigma]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} \mathbf{nil}}$$

$$\frac{\boxed{\langle \mathcal{I}, n, \kappa\rangle \rightsquigarrow (l, l', \mathbf{e})}}{l ::^{\delta} [\mathbf{eval}\,(Q)\,@l'.P]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} [P']^{\langle n',\kappa'\rangle} \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} [Q]^{\langle n,\kappa\rangle} \parallel l' ::^{\delta'} [P']^{\langle n',\kappa'\rangle}}$$

Figure 1. Semantics for acKlaim, the formalism for our system model. The important point is the reference monitor in the box, which evaluates whether a certain action is admissible in the system model. This test includes to check the action is performed from a neighbouring location and that the actor has the necessary credentials to perform the action.

the identity of an actor, as could be done by a face recognition, and access to both the server room and the user office are based on some cipher key. All locations in our model can also store data; naturally, computers would store virtual data, and rooms would store physical data. We omit most of the details of the semantics and formalisms of our model [PH08]. In the rest of this section we will only briefly sketch the analyses available on the model. In contrast to the approach discussed in this paper, those analyses are actor-centric. They identify possible attacks of actors onto the modelled organisation. In Section 3 we shift focus to data-centric applications.

- Conditional Reachability Analysis. Given a representation of the system, this analysis allows us to identify which places a user may reach, based on identity, knowledge and keys possessed, and location in the model. Such an analysis has two immediate applications; it enables us to identify possible shortcomings in an access-control system, and makes it possible to decide whom to use in order to reach a certain location or retrieve a certain piece of data.
- Log-trace Reachability Analysis. Like conditional reachability, this analysis receives a system model as input, this time with the logging extension [PH08], and a stream of logged events, for example recovered from some kind of logging system. This could either be the dump of logging units in the system, it could, however, also be observations made as part of an investigation, or a mix of both sources. Based on these, the analysis explores what an actor might have done in between two log entries. The more fine-grained the logging system is, the more precise the result of this analysis will be, but the more coarse-grained the logging system is, the more beneficial is our analysis. This is because the set of actions possibly performed between two log entries is getting bigger and bigger the further the two logged actions are apart. Consequently, it becomes harder and harder to keep a clear view of what might have happened in between.
- Online Surveillance. Using log-equivalent locations and observable log events, this analysis computes a finite representation that predicts the locations of actors based on observed actions in the logging system [PH09].

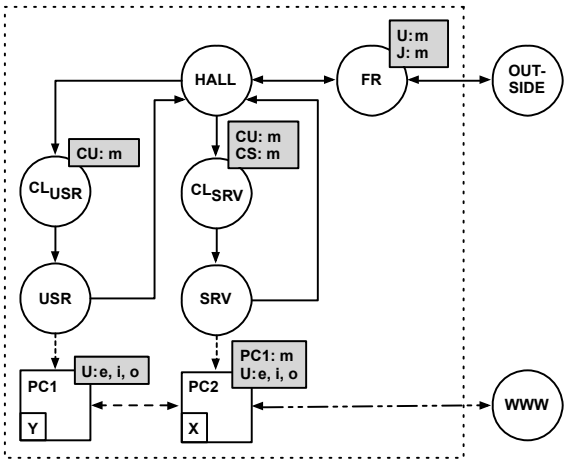

Figure 2. Example model for an organisation. The circles represent locations in the physical domain, such as a reception *REC* connected to the outside, a face recognition node *FR* that controls access to the hallway, etc. The gray boxes represent policies, *e.g.*, the face recognition node may only be entered by U and J, which are assumed to be the identities of actors. The white boxes represent locations (work stations) in the digital domain, and one of them is connected to the Internet, *e.g.*, to send email. The X and Y in the boxes represent data.

3. Tracing Provenance in System Models

In the work and the analyses described above we consider actors and their possible actions in the modelled organisation. In the work described in this article we consider reachability of *data* instead. In contrast to actors, data does not move on its own, and can not possess credentials required for fulfilling any policies. In acKlaim, data can only be moved by processes, which, as described above, represent either computer programs or actors.

The analysis for data provenance based on system models therefore tags data with possible ways through the system model. These paths can then be used to collect the policies applied, and the processes handling the data. For data tracing we have developed two analyses: a forward analysis, that is similar to the conditional reachability analysis described in Section 2, and a backward analysis, that identifies how data ended up in a certain location.

Like our earlier actor-centric analyses [PH08], the two analyses described in this section fall into two categories: the analysis of potential data targets predicts possible situations, and the analysis of data provenance starts from a given system state, and comutes possible paths of how data can have reached its location.

While the analyses presented in the rest of this section can potentially be expensive, they are not in real life. This is due to the fact that we compute unions on graphs, which can be reduced to

```
1: DataTargets
2: Input : system model S, start location L
3: for all locations n: procs(n) = ∅
4: worklist = L
5: while worklist not empty do
6:    pick ℓ from worklist
7:    ps = processes_DT(S, ℓ)
8:    for (s, r) ∈ ps do
9:       r' = procs(s) ∪ r
10:      if r' ≠ procs(s) then
11:         procs(s) = r'
12:         add s to worklist
13:      end if
14:   end for
15: end while
16: return
```

```
1: processes_DT
2: Input : system model S, location ℓ
                {(s, {u, p})|   process with an ID u at location s
3: R =                          can access data using policy p
                                from a neighbour of ℓ}
4: return R
```

Figure 3. Analysing the system model for potential targets for a data item. Starting from the initial location ℓ, the system model is explored by a work list algorithm. Whenever a node is visited, the algorithm determines who might take the data from here.

identification of strongly-connected components, and that typical policies are rather small, reducing the necessary iterations due to possible source of actions. What remains is the exploration of the graph.

3.1 Data Targets

The analysis that identifies potential targets for data items is similar to the one for conditional reachability described above. As mentioned before, data in our model can not move on its own, but must be moved by processes. In the actor-centric conditional reachability analysis [PH08] we trace actors based on their identity and owned data through the model.

In the analysis of potential data targets, we trace data through the model based on which processes can own or access it. While the former analysis only can discharge policies if the actor has the necessary credentials, this is not an issue in the data-centric analysis; since we are interested in legitimate data access, we assume that the process owning the data item is able to discharge policies. This is a valid abstraction as we are interested in which locations in the model the data might reach.

The algorithm shown in Figure 3 analyses a given system model and identifies the parts of the model that a data item can end up at. The algorithm is work-list based and terminates when no information of potential processes that can obtain data at a certain location changes. The main component is the function processes that returns sets of pairs where the first component is a location from which some process can obtain the data item, and the second component is a set containing the identity of the process being able to do so and the policies by which the access was permitted.

The algorithm starts from the location where a certain piece of data is located, *e.g.*, location *PC1* for data item *Y* in Figure 2. From this start location, the algorithm explores the system model. Whenever there is a neighbouring location that may access the location being analysed, that location is added to the worklist.

If we consider the data item *Y* located at *PC1* in the example shown in Figure 2, then the work list algorithm starts investigating *PC1*. The possible pairs returned by processes are

- $(\mathrm{USR}, \{\mathrm{U}, [U : i]\})$ and $(\mathrm{PC2}, \{\mathrm{U}, [U : i]\})$, since the user U is allowed to input at *PC1* from the two neighbouring locations *USR* and *PC2*,
- $(\mathrm{USR}, \{\mathrm{U}, [U : e]\})$ and $(\mathrm{PC2}, \{\mathrm{U}, [U : e]\})$, since the data might be accessed by a process started by user U at *PC1* from the two neighbouring locations, and
- $(\mathrm{PC1}, \{\mathrm{U}, [U : i]\})$ since a process executing at *PC1* could access the data.

As a result of this inspection, the work list is extended with the locations *USR*, *PC1*, and *PC2*. Eventually, the computation results in a fixpoint, where for example the result for the location *OUTSIDE* is $(\mathit{PC1}, U, [U : i]), (\mathit{FR}, U, [U : m])$ representing that the user input the data and then moved through the node *FR*, or $(\mathit{PC2}, U, [U : e]), (\mathit{PC2}, U, [U : i]), (\mathit{FR}, U, [U : m])$, representing that the user started a process, input the data, and then moved.

3.2 Data Provenance

The analysis for identifying data targets presented in the previous section is a forward analysis that traces data from its original location to those locations, which that data item potentially can reach. We now present a backward analysis, that finds out where data actually may come from.

The data provenance analysis is similar to the log-based analysis in ExASyM [PH08]; based on the assumption that a data item is observed at a certain location in the model, the analysis finds out how that item can have reached that location from its original location. This is also similar to approaches for invalidating policies [KP13], where data is not allowed to reach a certain location.

Like the analysis for identifying data targets, also the data provenance analysis does assume that actions are not blocked by policies. As discussed above this assumption makes sense since we assume an actor to have handled the data. We have in earlier work developed analyses for the case of illegitimate actions [PHN07], which can be trapped by the reference monitors show in Figure 1.

The analysis for computing data provenance based on system models is presented in Figure 4. The arguments to the analysis are a system model, the location where data has been observed (or will be used), and location where the data is coming from originally. Also this algorithm is work-list based; it explores the system model and terminates when the data item's original source location is reached. In contrast to the (forward) analysis for identifying potential data locations, we now look at *incoming* edges at the inspected location. The core of the analysis method is the same as in the data targets method, just the selection of neighbouring nodes to inspect is different.

If we consider that the data item X is located at location *WWW*, then we need to simulate how the data might have ended up there. While we do not consider the underlying formalism's semantics here, the data can only have been output there, or must have come with a process; actually, the latter holds in any case since the data must have come to the location before being output. To reach the *WWW* location, the data must either have come with a process, or it must have been output by a process at *PC2*. If we consider the data item *Y*, then it can either have come with a process from *PC2* and then with a process from *PC1*, or a process at *PC1* must have output the data at *PC2*.

Beyond the selection of nodes and the exploration of the graph the two analyses are almost identical, so we skip the rest of the algorithm. We only note that both algorithms can, *e.g.*, be extended with a component to take observed actions from a log file into

```
1: DataProvenance
2: Input : system model S, goal location G, data location D
3: for all locations n: procs(n) = ∅
4: worklist = G
5: while worklist not empty do
6:    pick ℓ from worklist
7:    ps = processes_DP(S, ℓ)
8:    for (s, r) ∈ ps do
9:       r' = procs(s) ∪ r
10:      if r' ≠ procs(s) then
11:         procs(s) = r'
12:         if s == D then
13:            continue
14:         end if
15:         add s to worklist
16:      end if
17:   end for
18: end while
19: return
```

```
1: processes_DP
2: Input : system model S, location ℓ
3: R = {(s, {u, p}) | process with an ID u at location s can output data or bring data using policy p from a predecessor of ℓ}
4: return R
```

Figure 4. Analysing the system model for potential sources of data items. Starting from the initial location $\mathcal{S}$, where the data has been observed, the system model is explored by a work list algorithm. The algorithm traces the data item back to all possible locations.

account to narrow down what really has happened in the modelled organisation, and to investigate the relation between data flow and logged actions. We have earlier introduced the notion of log-equivalence [PH08], which describes all actions that can be performed stealthy between observed actions. This concept can easily be extended to cover data as well.

4. Conclusion and Future Work

Identifying provenance of data provides insights to the origin of data and intermediate results, and has recently gained increased interest due to data-centric applications. In this work we extend a data-centric view with actors handling the data and policies restricting actions.

Data provenance is an important question to be answered in many scenarios. While typical applications to provenance mainly consider processes, we discuss in this work how to complement existing approaches with support for spatial-organisational and socio-technical aspects. This complementation is based on analyses that are data-centric as opposed to the actor-centric analysis in system models.

The extension of provenance to models allows us to trace data and its provenance beyond software, process, and workflow boundaries. This extension can also enable dynamic surveillance methods like the ones discussed above for ExASym, where sensors in the organisation can trace data.

We believe that the combination of system models, analyses, and data provenance enables new insights into the flow of data, how to avoid inadmissible data flow, etc. We also explore the application of our analysis to smart grid security; a potential application of data provenance together with infrastructure models is to identify potential sources of malware.

We are currently extending our system model to cover also workflows, where the data-centric approach immediately is applicable, and we are working on a prototype implementation where our model is extended with the analyses presented in this paper.

Acknowledgment

Part of the research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TRE$_{\text{S}}$PASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.

References

[BLP02] Lorenzo Bettini, Michele Loreti, and Rosario Pugliese. An infrastructure language for open nets. In *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, pages 373–377, New York, NY, USA, 2002. ACM.

[dNFP98] Rocco de Nicola, Gian Luigi Ferrari, and Rosario Pugliese. KLAIM: A kernel language for agents interaction and mobility. *IEEE Trans. Softw. Eng.*, 24(5):315–330, May 1998.

[DPH10] Trajce Dimkov, Wolter Pieters, and Pieter Hartel. Portunes: representing attack scenarios spanning through the physical, digital and social domain. In *Proceedings of the 2010 joint conference on Automated reasoning for security protocol analysis and issues in the theory of security*, pages 112–129. Springer, 2010.

[GP03] Daniele Gorla and Rosario Pugliese. Resource access and mobility control with dynamic privileges acquisition. In Jos C.M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 119–132. Springer Berlin Heidelberg, 2003.

[KP13] Florian Kammüller and Christian W. Probst. Invalidating policies using structural information. In *2nd International IEEE Workshop on Research on Insider Threats (WRIT'13)*. IEEE, 2013. Co-located with IEEE CS Security and Privacy 2013.

[PH08] Christian W. Probst and René Rydhof Hansen. An extensible analysable system model. *Information Security Technical Report*, 13(4):235–246, November 2008.

[PH09] Christian W. Probst and René Rydhof Hansen. Analysing access control specifications. *2009 Fourth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pages 5,6, 2009.

[PHN07] Christian W. Probst, René Rydhof Hansen, and Flemming Nielson. Where can an insider attack? In *Proceedings of the 4th international conference on Formal aspects in security and trust*, FAST'06, pages 127–142, Berlin, Heidelberg, 2007. Springer-Verlag.

[Pie11] Wolter Pieters. Representing humans in system security models: An actor-network approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):71–88, 2011.