

# On Information Flow Forensics in Business Application Scenarios

Claus Wonnemann, Rafael Accorsi, and Günter Müller  
*Department of Telematics*  
*University of Freiburg, Germany*  
{wonnemann, accorsi, mueller}@iig.uni-freiburg.de

## Abstract

*To-date, security analysis techniques focus on the explicit access to data, thereby neglecting information flows happening over covert channels. As a result, critical business software applications and their deployment may be labeled secure, whereas in fact they are not. We present ongoing research towards information flow forensics, a novel approach for the a-posteriori detection of information flow. We motivate our work by illustrating the implications of illicit information flow in different software application scenarios and demonstrate why current approaches fall short of effectively enforcing information flow policies in many cases. We show that information flow forensics can mitigate these drawbacks and outline some interesting research challenges involved in its realization.*

## 1. Introduction

Security and privacy play a central role in virtually any software application used in digital business, such as databases, web browsers or implementations of automated business processes.

Generally, a software application must comply with *security policies*. These policies express non-functional requirements, such as the confidentiality of processed data and constraints or obligations put on its usage. Their enforcement relies on secure design methodologies and on reference monitors. While these enforcement mechanisms can effectively prevent illicit accesses to data, e.g. when user *A* illegally requests access rights to object *X*, they can neither control nor recognize the so-called *covert channels* [1], i.e. channels that exploit a mechanism whose primary purpose is not information transfer, but which is misused for it. Covert channels, such as timing channels and implicit flows [2], can lead to unintended information flows (IF) and, eventually, policy violations (see Section 2). As a result, an application and its deployment could

be labeled “policy compliant,” whereas a thorough analysis of its IF would in fact reveal the opposite.

While access control for digital business applications is well-studied and standardized (e.g. [3], [4], [5]), as we show in Section 3, existing approaches to IF control (IFC) based on static analysis cannot cope with this setting. Our thesis is that, albeit neglected until now, IF pose a great threat to policy compliance and must be equally considered. In fact, as the examples in Section 2 illustrate, without the consideration of IF, no strong compliance guarantees can be given.

We report on ongoing work towards *information flow forensics* (IFF), a novel approach for a posteriori detection of IF in software applications [6]. The idea of IFF proposes the a-posteriori detection of IF through inspection of authentic log data, thereby making IF analysis an audit task. Clearly, forensic analysis cannot prevent illicit flows. However, it enables their timely recognition and the initiation of countermeasures.

At the heart of IFF is the (“intelligent”) analysis of log data collected during the software execution. Other than a trivial audit based on pattern matching, such an analysis must consider and establish the (potential) correlations and dependencies of (clusters of) events at different abstraction levels, and test these emerging patterns against IF policies. Overall, IFF opens up a number of interesting research challenges, some of which are described in Section 4.

This paper makes thus the following contributions:

- It introduces IFF approach to detecting IF in business applications, thereby focusing on the research challenges involved. The forensic viewpoint is to our knowledge new and can lead to the reliable deployment of business applications and advanced analysis and audit techniques.
- It motivates the need for analysis techniques by presenting several practical scenarios and shows the implications IF can cause.
- It discusses the scope and limits of static, preventive approaches to IFC in business applications.

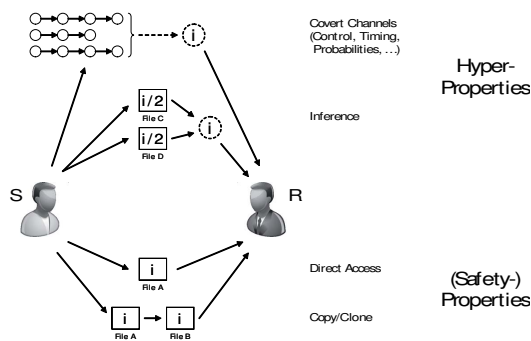


Figure 1. Types of information flow from S to R.

## 2. Security Threats of Information Flow

Figure 1 shows the different ways to transmit a piece of information  $i$  from sender  $S$  to recipient  $R$ . The lower half of the figure depicts the so-called “direct” transfers of information by retrieving the value of some data container, e.g. variable or database. These transfers are called *accesses*. The upper half of the figure shows the more subtle, “indirect” transfers through observation and inference. These transfers are called *information flows*, or simply IF.

IF are caused by all sorts of variations in the behavior of a system from which information can be extracted. Figure 2 illustrates a situation during the execution of an automated business process (i.e. the ordered invocation of service applications to reach a business goal) in which threats from illicit IF through covert channels may arise. At one point during process execution, sensitive data is processed or confidential events occur. Subsequent steps in the process provide an interface to an outside party, which can observe process events or processed data. If the “secret” compartment of the process influences its “public” parts, for instance by causing variations in the time-lag between the occurrence of public events, an outsider can possibly deduce secret information.

While IF is largely neglected when software is written and deployed, it poses a grand threat for the confidentiality of sensitive data in most applications. For instance, by exploiting the fact that the response times of many applications differ significantly depending on whether a provided username is valid or not, it is possible to verify the existence of account names [7]. Further research has shown that more detailed information, such as the size of data sets and even the private keys of RSA crypto systems, can be obtained through timing attacks in distributed applications [8].

Besides timing, there are several other types of covert channels including [2]:

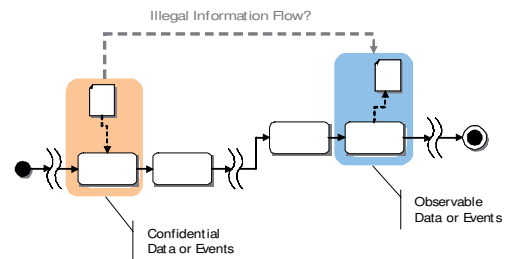


Figure 2. The IF control challenge for e-business processes.

- *Implicit flows* signal information through the control flow of a system.
- *Power channels* enable an attacker to deduce information from the energy consumption of a device (such as RFID transponders or smart cards).
- *Termination channels* leak information through the termination or nontermination of an execution.
- *Resource exhaustion channels* signal information through the possible exhaustion of a finite shared resource, such as disk space or memory.
- *Probabilistic channels* transmit information through changes in data’s probability distribution.

Usual ways to express IF properties are those specifying the entropy of data [9], as well as the Biba policy [10] and the various ways of expressing IF policies based on the original notion of noninterference [11]. Overall, while these policies still express safety requirements – there must be no illegal information transfer –, they are generally *not* properties in the classical sense. For in order to decide an IF policy, one must consider sets of sets of executions. Therefore, IF policies denote the so-called *hyperproperties* [12].

### 2.1. Scenarios

Information flow poses a great challenge in several fields of digital business [2]. The following lists four scenarios in which thorough IF analysis is necessary.

*Database queries.* Database queries encompass both accesses and IF. By answering to a legitimate query, a subject obtains access to the set of tuples satisfying the query. However, these tuples may also release data that allows the (illegitimate) inference of data items. In particular, successive, targeted queries can decrease the entropy of variables to the point that its precise value can be inferred.

This is a threat to compliance requirements demanding confidentiality of data, such as the HIPAA privacy rule [13]: A patient’s privacy is at risk when anonymized pieces of data are released to third parties,

which might accumulate several of these pieces so that patients' identity (and other pieces of information) are eventually inferred [14].

*Web browsers.* Web browsers offer an extensible, customizable platform, on top of which a number of applications may run in parallel. Here, two issues arise. The first is *tabbing*, an ubiquitous feature of modern web browsers that allow users run the applications embedded into different web pages in multiple tabs, e.g. an e-banking site or a (less trusted) blog page. Depending on the contents of the web pages, covert channels may be built from one tab to the other, using to this end the kernel of the web browser or other system resources to transmit information. As a result, confidentiality and privacy policies may be violated.

The second are *feature interactions*. Mozilla, for example, can be extended with thousands of "add-ons" that add functionalities (i.e. features) to the browser kernel for, e.g., improved appearance, (interactive) web development, social networking, and enhanced security. While add-ons have a positive impact on the usability of browsers, they open up the possibility of feature interactions, i.e. interactions between the plugins and the browser kernel that modify the behavior of the overall system in unpredictable ways [15]. One possible consequence of feature interactions can be covert channels that can be observed by an intruder.

*Online e-voting.* While the use of voting machines has been largely discouraged and in some countries, such as Germany and France, even completely forbidden, the use of online e-voting in small-scale elections is rapidly gaining on momentum. In this setting, time channels may happen in the e-voting application. As an example consider a voting system with two options, in which the system behaves differently depending on whether the first or the second option has been chosen. If, for instance, further processing takes significantly more time for option 1 than for option 2, just by observing the processing time, an intruder may draw conclusions on which option was chosen (here, the vote is signaled through a time channel), as well as linking the option with the voter.

Besides time channels and possibly other forms of covert channels, small-scale e-voting also allows inference of votes [16], specially in cases where the tendency of some voters are known to an observer or to the entity counting the ballots.

*Supply chain management.* Supply chain management is the management of a network of interconnected businesses involved in the provision of product and service packages. In this setting, the allocation and

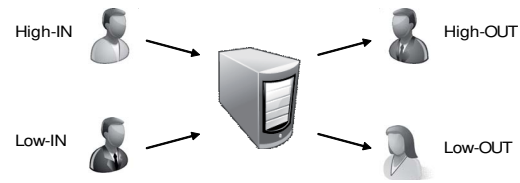


Figure 3. Classified inputs and outputs of a system.

distribution of assembly parts and components of a product are of decisive strategic value for a company and must thereby be kept secret. Keeping this information confidential becomes more complicated, as a producer may serve several enterprises.

ERP modules automate most of this processes and spies observing the control flow of the business processes, the time necessary to fulfill a task and its termination may be in a position to deduce the whole – or at least parts of – the supply chain.

### 3. Information Flow Enforcement

While the enforcement of access control policies is straightforward (using an inline execution monitor), noninterference and related policies cannot be enforced with runtime techniques alone. In order to decide whether two data containers interfere in some way (e.g. through covert channels), a security mechanism needs a broader knowledge of the target system than an execution monitor usually has. By definition, an execution monitor's knowledge of the target system is restricted to the execution observed by the monitor; such policies, which can be decided by looking at single executions only, have been coined *properties* [17].

To decide whether two data containers interfere in some way, e.g. over covert channels, it is not sufficient to observe single executions. Noninterference, for example, stipulates that "high" (i.e. confidential) actions must have no effect on system behavior that can be observed by "low" (i.e. untrusted) users (see Figure 3). This rationale is formalized by requiring that there must always be multiple system executions generating the same observable system behavior, so that an outside observer cannot decide whether some "high" actions were involved [18]. Thus, whether a certain system execution is allowed or not depends on whether *another* execution is allowed. Hence, to enforce such a policy a monitor would need knowledge of multiple executions, which could per se generate control flows. Technically, this is a consequence of the fact that IF policies denote hyperproperties.

In consequence, IF policies (and hyperproperties in

general) are enforced before the fact by static analysis only.<sup>1</sup> A wide range of techniques has been developed for the analysis of IF policies. These analysis techniques can be distinguished between those analyzing abstractions or models of software applications [20], [21], those analyzing the source code of software applications [22], [23], and those embedded in the programming language (so-called “language-based security” [2]), thereby enforcing some IF policies from the outset [24], [25].

### 3.1. Drawbacks of Static Techniques

Static enforcement of IF requires to specify the target system in a tool-specific notation (such as a special programming language or the description logic of a theorem prover) and yields the decision whether the system respects a given policy or not. This approach is feasible only for niche applications, but is in most cases impractical for the following reasons:

- **Inflexibility of policies.** As laws and regulations change over time, policies governing computer systems must evolve. Static approaches yield systems that are tied to a specific policy, defined at design time. Policy change, e.g. to rule out some IF that was legitimate before, requires to change the system itself. This requires substantially more effort than modifying the access control lists of a runtime monitor, for instance.
- **Implementation-dependent IF.** Static analysis cannot reason about IF that depends on the context in which an application runs, such as its timing behavior. In most cases, the scope of current techniques is therefore restricted to implicit IF.
- **Existence of legacy software.** The verification of existing software usually requires its re-specification for a certain tool, virtually amounting to re-writing the entire code.
- **Lacking modularity.** IF in general does not compose, i.e. there is no guarantee that the composition of IF-compliant components is compliant. There might be emergent behavior leading to policy violations.

As a result, current IFC techniques are virtually not used outside the academic world [26]. For the reasons mentioned above, they seem to cumbersome to use for most modular applications, that underlie fast-moving development cycles and that are subject to shifting security requirements.

1. Recent work shows that dynamic dependency monitoring can increase the precision of static analysis [19].

## 4. Information Flow Forensics

To complement static IF analysis, we investigate the *a posteriori* detection of IF in a manner similar to forensic investigations. For this reason, we coined the term “information flow forensics.” Building on authentic log files recorded during execution of the target system, the goal is to advance IF control by developing approaches for the analysis of log data to detect illegal IF. In doing so, illegal IF cannot be prevented; instead, its detection is supported, making illegal IF accountable and enabling countermeasures.

Compared to static verification, IFF is directly applicable to existing systems, as it relies on log data which is generated by virtually any application software. Further, the policies governing the analysis can be modified without changing the system itself. Since log data reflects the system behavior in its actual deployment environment, implementation-dependent IF (particularly timing channels) can be discovered, effectively expanding the scope of IF analysis.

Provided with a policy and log data, an audit mechanism is either to return the proof that no illicit IF with respect to a policy has taken place (on the policy’s abstraction level), or to give evidence for policy violations and their circumstances (for instance, the type of IF, its locations and cause).

We currently investigate the following issues:

- **A system model for forensic analysis.** Research on IFC has developed a variety of information flow policies, addressing different security requirements. One particular well-studied class, the so-called possibilistic security properties [27], [28], captures only implicit IF, for instance. We are currently investigating a system model in which policies for essential types of IF (including timing and probability) can be specified and which is sufficiently abstract to be mapped to different target applications, such as business process implementations and web browsers.
- **Data selection.** The basis for the analysis is log data. While mechanisms for secure logging exist, to-date it is unclear which pieces of information are in fact relevant for the detection of illegal IF. One of our efforts is thus to select the log data to be collected for the analysis.
- **Development of analysis algorithms.** We develop audit mechanisms for IFF to decide whether a set of audit trails contains illicit IF in terms of a specified policy. Efficiency is a particular challenge, since – IF policies being hyperproperties – *sets* of execution traces in possibly large pools of log data must be analyzed. The intelligent

analysis techniques we aim at employing allow for a precise analysis of audit trails.

Other issues that remain to be investigated include the selection of log data to feed the analysis engine and the certainty of generated evidence – are there false negatives or positives? –, which is much a function of the underlying analysis technique.

## 5. Conclusion

Information flow poses a major threat to security and compliance requirements of business applications. With access control monitors usually being the only security mechanisms, IF arising from covert channels and from information inference may go undetected.

This paper makes the case for the a-posteriori detection of illegal IF. We firmly believe that the detective approach to information flow analysis offers enormous advantages in the context of business application software. Besides its practical potential, we have identified a number of scientific challenges to be addressed in order to make such forensic analysis feasible, thereby eventually improving the toolset of audit techniques.

## References

- [1] B. W. Lampson, “A note on the confinement problem,” *Comm. the ACM*, vol. 16, no. 10, pp. 613–615, 1973.
- [2] A. Sabelfeld and A. C. Myers, “Language-based information-flow security,” *IEEE J-SAC*, vol. 21, no. 1, pp. 5–19, 2003.
- [3] H. Koshutanski and F. Massacci, “An access control framework for business processes for web services,” in *ACM workshop on XML security*. 2003, pp. 15–24.
- [4] R. Sandhu and P. Samarati, “Access control: Principles and practice,” *IEEE Comm. Mag.*, vol. 32, no. 9, pp. 40–48, 1994.
- [5] E. G. Sireer and K. Wang, “An access control language for web services,” in *ACM SACMAT*. 2002, pp. 23–30.
- [6] R. Accorsi and C. Wonnemann, “Detective information flow analysis for business processes,” in *Business Processes, Services Computing and Intelligent Service Management*, ser. LNI, vol. 147, 2009, pp. 223–224.
- [7] A. Bortz, D. Boneh, and P. Nandy, “Exposing private information by timing web applications,” in *Conf. World Wide Web*. 2007, pp. 621–628.
- [8] D. Brumley and D. Boneh, “Remote timing attacks are practical,” *Comp. Net.*, vol. 48, no. 5, pp. 701–716, 2005.
- [9] L. Sweeney, “k-Anonymity: A Model for Protecting Privacy,” *J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.
- [10] K. J. Biba, “Integrity considerations for secure computer systems,” The Mitre Corp., Tech. Rep., Apr. 1977.
- [11] J. A. Goguen and J. Meseguer, “Security policies and security models,” in *IEEE SSP*, 1982, pp. 11–20.
- [12] M. R. Clarkson and F. B. Schneider, “Hyperproperties,” in *IEEE CSF*. 2008, pp. 51–65.
- [13] “HIPAA: Health Insurance Portability and Accountability Act,” <http://www.cms.hhs.gov/HIPAAGenInfo/>.
- [14] R. Accorsi, Y. Sato, and S. Kai, “Compliance monitor for early warning risk determination,” *Wirtschaftsinformatik*, vol. 50, no. 5, October 2008 pp. 375–382.
- [15] P. Zave, “FAQ Sheet on Feature Interaction,” <http://www.research.att.com/pamela/faq.html>, 2004.
- [16] T. Endo, I. Echizen, and H. Yoshiura, “Electronic voting scheme to maintain anonymity in small-scale election by hiding the number of votes,” in *ARES Conference*. 2008, pp. 1287–1293.
- [17] F. Schneider, “Enforceable Security Policies,” *ACM TISSEC*, vol. 3, no. 1, pp. 30–50, 2000.
- [18] J. Mclean, “Security models,” in *Encyclopedia of Software Engineering*. Wiley and Sons, 1994.
- [19] P. Shroff, S. F. Smith, and M. Thober, “Dynamic dependency monitoring to secure information flow,” in *IEEE CSF*, 2007, pp. 203–217.
- [20] Á. Darvas, R. Hähnle, and D. Sands, “A theorem proving approach to analysis of secure information flow,” *SPC*, ser. LNCS, vol. 3450. 2005, pp. 193–209.
- [21] T. Amtoft and A. Banerjee, “Verification condition generation for conditional information flow,” in *ACM FMSE*. 2007, pp. 2–11.
- [22] D. Volpano, C. Irvine, and G. Smith, “A sound type system for secure flow analysis,” *JCS*, vol. 4, no. 2-3, pp. 167–187, 1996.
- [23] C. Hammer, J. Krinke, and G. Snelling, “Information flow control for java based on path conditions in dependence graphs,” in *IEEE ISSSE*. 2006, pp. 87–96.
- [24] F. Pottier and V. Simonet, “Information flow inference for ML,” in *ACM TOPLAS*, vol. 37. 2002, pp. 319–330.
- [25] A. Myers, N. Nystrom, L. Zheng, and S. Zdancewic, “Jif: Java + information flow,” 2001.
- [26] S. Zdancewic, “Challenges for information-flow security,” *PLID*, 2004.
- [27] A. Zakinthinos and E. S. Lee, “A general theory of security properties,” in *IEEE SSP*, 1997, pp. 94–102.
- [28] H. Mantel, “Possibilistic Definitions of Security – An Assembly Kit,” in *IEEE CSFW*. 2000, pp. 185–199.