Why Tags Could be It[⋆]

Keynote Lecture
Extended Abstract

Fred B. Schneider[⋆⋆]

Department of Computer Science
Cornell University
Ithaca, New York 14853 USA
`fbs@cs.cornell.edu`

Abstract. Reference monitors embody specifications about permitted and prohibited operation invocations. That limits what policies they can enforce. Those limitations have prompted us to explore alternative approaches to policy enforcement—specifically, expressive classes of labels that give permitted and prohibited uses for a piece of information. These *reactive information flow* (RIF) labels will be described, along with means for static and run-time verification of programs that process such labelled data. Use of RIF labels for specifying use-based privacy also will be discussed.

1 Introduction

Security policies can be enforced by defining guards on operations or by associating labels with values, as follows.

- A *guard* on an operation *Op* is checked each time *Op* is invoked; the guard blocks any invocation that would not comply with the policy.
- A *security label* on a value or variable V is checked before V is read or written; the access is blocked when it is inconsistent with what the security label allows.

Today's systems tend to be built in terms of guards on operations rather than in terms of security labels on values. This is unfortunate, because security labels specify and provide end-to-end guarantees about information use, whereas guards on operations do not.

For example, consider a system that creates and maintains a replica F' of some file F. A guard that prevented principal *Alice* from invoking a `read` operation naming F is not obliged to prevent *Alice* from invoking a `read` operation

[⋆] Joint work with Cornell Ph.D. students Elisavet Kozyri and Eleanor Birrell.

[⋆⋆] Supported in part by AFOSR grant F9550-16-0250 and grants from Microsoft. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the U.S. Government.

naming F'. But an end-to-end guarantee that stipulates *Alice* not read the contents in F would have to prevent attempts by *Alice* to learn the contents of F' or other values derived directly or indirectly from the contents in F. In addition, security tags can afford providers of information with flexibility to choose security policies after a system has been developed, deployed, or put into operation. Policy now accompanies a system's inputs instead of being fixed in the code.

2 Reactive Information Flow Specifications

The prevalence today of guards over security labels is not surprising, given limitations in the expressive power of currently available classes of security labels. To help overcome those limitations, we have been developing a new class of security labels: *reactive information flow* (RIF) specifications. Informally, a RIF specification for a value V gives

(i) allowed uses for V, and
(ii) the RIF specification for any value that might be directly or indirectly derived from V.

RIF specifications thus give allowed uses for the value produced by evaluating a function, where those restrictions may differ from the allowed uses for inputs to that evaluation. For instance, using RIF specifications as labels, the output of an encryption function can be public even though is inputs (plaintext and a key) are secret. In general, RIF specifications support *reclassifiers* that increase restrictions, decrease restrictions, or associate incomparable restrictions.

Various *carriers* can be instantiated to embody RIF specifications. A carrier must accept a language of reclassifiers, and it must associate a set of restrictions with each word in that language. Carriers for which language-inclusion is decidable are a good choice when we wish to treat RIF specifications as types, since the resulting type system will be statically checkable. To date, we have experience with two classes of (decidable) carriers.

– Finite state automata suffice for many common security needs. Here, each automaton state gives a set of use restrictions; reclassifiers label transitions between automaton states, with the successor automaton state giving the new set of use restrictions for a derived value.
– A simple form of push-down automata suffice for handling confidentiality when encryption and decryption are used to transform values (typically from secret to public and back). Encryption pushes a key onto the stack; decryption causes pop if the key being provided matches the key contained in top entry on the stack (and otherwise the decryption causes a push).

Type systems have been formulated for both kinds of carriers, where type correctness ensures that certain non-interference properties are satisfied. The conservative nature of type checking, however, is now leading us to contemplate run-time monitors for programs having RIF specifications as labels for values and variables. We also have been exploring practical aspects of using RIF specifications. For this, the information-flow type system in the JIF programming language has been replaced by a RIF type system based on finite-state automata. Prototype applications that we programmed in this JRIF language have given us experience with defining RIF specifications.

3 What RIF Tags May Restrict

Security labels traditionally have been interpreted as characterizing sets of principals. For confidentiality, a label specifies principals that are allowed to read a value (or any value derived); for integrity, a label describes principals that must be trusted for the labeled value to be trusted (which implies that the label defines a set of principals that may update the labeled value).

In practice, other forms of use restrictions are important too. In *use-based security*, pieces of information are labeled—actually or notionally—with tags that specify *use restrictions*, and principals who hold or process such pieces of information are obliged to comply with those restrictions. Use restrictions may come from those who submit or control the information, systems that process the information, and/or regulations imposed by the jurisdiction in which a system is located, the data originates, or its owners reside.

Use-based security can be quite general if we are given an expressive enough language for specifying the use restrictions. By choosing a suitable language, for example, we can support the various definitions of privacy that are being discussed, now that the failings of classical "notice and consent" have become apparent. We can also support regimes where data collection and use are limited by legislative authorities that specify when and how data may used, combined, how long it must be saved, etc.

RIF specifications seem well suited for defining restrictions for use-based security. Here, restrictions are not limited to being sets of principals; the restrictions instead can be permissions, prohibitions, and/or obligations for invoking arbitrary classes of operations. Reclassifiers, as before, allow derived values to be subject to different use restrictions. This capability, for example, would enable a RIF specification to assert that an individual's value must be kept confidential, but any derived value produced by statistical aggregation is public.

4 Enforcement

Formal verification, automated analysis, and run-time monitoring all are time-honored methods to ensure that a program will satisfy some property of interest. The trade-offs between expressiveness, conservatism, and automation are likely to be the same for RIF specifications as has been found for other classes of program properties. In connection with privacy, however, audit, with deterrence through accountability is sensible. So instead of preventing violations, a system detects violations and recovers. Prevention is not necessary, here.