Service Automata for Secure Distributed Systems*

Poster Proposal

Richard Gay Heiko Mantel Barbara Sprick

CASED/TU Darmstadt, Germany

{gay,mantel,sprick}@mais.informatik.tu-darmstadt.de

Secure Distributed Systems. Together with the rising popularity and ubiquity of flexible distributed systems paradigms, such as service-oriented architectures and clouds, also the demand for equally flexible security solutions increases. However, one cannot easily transfer established security solutions for stand-alone systems to distributed systems, just as the security of distributed systems cannot be reduced to the security of each its components or the security of a single large monolithic system.

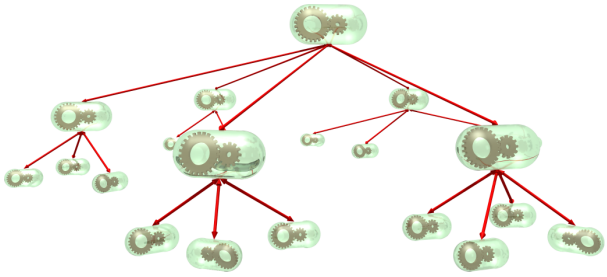
Consider the example of a company providing software repository storage to other companies in the form of services. Even if we assume that every single service consists of a perfectly secured repository server, we may not conclude that these services fulfill the security requirements of the customers: the latter would probably desire that no user of the services may ever obtain or even manipulate the repository content of two competing companies, even if the user might have the permission to access each of the two repositories alone. Here, local access control does not suffice but needs to be supplemented by some kind of state exchange between services.

Runtime Enforcement. A generic approach for ensuring the adherence of programs to security policies is enforcing them at runtime. For that purpose, a target program is monitored while executing and interventions are performed by the mechanism right before a violation of the given security policy is about to occur. This does not require for security analyses prior to the use of the program and, hence, allows for a flexible selection of the security policy based on the users' demands. The power of runtime enforcement in general is well-understood [5, 4, 3]. For instance, it has recently been shown how to provably enforce separation of duty properties with the help of runtime enforcement [1]. This approach and runtime enforcement in general however cannot be not easily applied to a distributed system: a naïve try could, e.g., introduce a central state-keeping component to be queried whenever a node in the system wants to execute a security-relevant action. However, such a central component potentially effects a performance bottleneck and single point of failure.

*This work is supported by CASED (www.cased.de)

Service Automata. Our goal is to provide a runtime enforcement architecture that offers substantiated security guarantees for distributed systems, while at the same time not introducing bottlenecks or limiting the flexibility of service users and service providers. The provided security guarantees may target the service users, whose data are protected from misuse, and/or the service providers, whose services shall not be exploited by their users.

Our architecture will be built upon *service automata*: local controllers for services, which communicate with each other in a hierarchically structured (see figure to the right) and efficient fashion to exchange security state information. These controllers shall act like encapsulations around services that protect the services from potential harm by the environment and the environment from potential harm by the service. They are parametric in the security policy to be enforced, which makes the architecture deployable easily and flexibly. At the same time, our architecture does not premise certain implementation details of the services, as it is envisioned to be a generic approach applicable to a wide range of settings, scenarios and security demands.



Service Automata Architecture

Enforcing Chinese Wall Policies. We have already developed formal specifications of service automata which are capable of enforcing, e.g., Chinese Wall security policies [2]. In terms of the scenario depicted above, no user of the repository services would ever be permitted to access data from competing companies, if the services were protected by our service automata. In the above figure, repository services would be the gears on the lowest level of the hierarchy while the superordinate levels are state-keeping and decision-making components.

Future Plans. Firstly, we strive to extend the class of security policies provably enforceable by our service automata and thereby widen the set of target application scenarios. Secondly, we are also searching for ways to increase the efficiency of the service automata, for instance by further reducing the communication and state-keeping requirements. Finally, while our current work aims at providing the foundations for service automata, we also implement our service automata in order to evaluate their practicability and efficiency.

References.

[1] D. A. Basin, S. J. Burri, and G. Karjoth. Dynamic Enforcement of Abstract Separation of Duty Constraints. In *14th ESORICS*, LNCS 5789, pages 250–267. Springer, 2009.

[2] D. F. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *IEEE S&P*, 1989.

[3] K. W. Hamlen, G. Morrisett, and F. B. Schneider. Computability Classes for Enforcement Mechanisms. *ACM TOPLAS*, 28(1):175–205, 2006.

[4] J. Ligatti, L. Bauer, and D. Walker. Edit Automata: Enforcement Mechanisms for Run-time Security Policies. *IJIS*, 4(1–2):2–16, 2005.

[5] F. B. Schneider. Enforceable Security Policies. *ACM TISSEC*, 3(1):30–50, 2000.