Flooding and Recycling Authorizations

Konstantin Beznosov
Laboratory for Education and Research In Secure Systems Engineering
University of British Columbia

beznosov@ece.ubc.ca

Abstract: The request-response paradigm used for access control solutions commonly leads to point-to-point (PTP) architectures with security enforcement logic obtaining decisions from the authorization servers through remote procedure calls. In massive-scale and complex enterprises, PTP authorization architectures result in fragile and sub-efficient solutions. The architectures also fail to exploit virtually free CPU and network bandwidth resources. This paper proposes to leverage publish-subscribe architectures for increasing failure resilience and efficiency by flooding delivery channels with speculatively pre-computed authorizations and recycling them on just-in-time basis.

1 Introduction

The departure point of our approach is based on a three-fold view. First, we observe that the microprocessor technologies evolved to the point where the processor resources are virtually free. Componentization of personal computers drove the market into the current state where commodity computing became the most cost-effective, forcing specialized computer architectures, operating systems, and solutions out of the market, with large organizations steadily moving towards commodity OSs (e.g., Windows and Linux) running on farms of PCs. Network bandwidth follows same path albeit slower. Yet, the cost of human time and attention seems to continue increasing.

Given the above context, this paper revisits the traditional approach for authorization architectures and puts forward quite a different view consisting of three key elements. First, it suggests decoupling the enforcement and decision parts of access control solutions with a publish-subscribe architecture. As a result, we expect the administrative and operating overheads associated with reconfiguring access control systems to accommodate component and infrastructure failures to be reduced. Then, since multiple EFs can subscribe to and share same authorizations, we are developing a means for

"recycling" authorizations by searching through the authorization streams and caches and finding the "best approximate" authorizations for the request in question. Last but not least, we consider taking the advantage of virtually free CPU and bandwidth resources by speculatively pre-computing authorizations, and flooding the publish-subscribe channels with them.

The expected results of this approach are: 1) better resilience of the authorization infrastructures in the face of high failure rates that can be observed in massive-scale enterprises (i.e., with computer populations on the order of hundreds of thousands); 2) reduced latency accounted for requesting and obtaining authorizations, and 3) reduced human resources required to maintain massive-scale enterprises.

The main contribution of this paper is in suggesting the three related approaches of publish-subscribe, authorization recycling, and flooding EFs with speculatively pre-computed authorizations as a promising alternative to the state of the art in authorization infrastructures, which are based on RPC and PTP.

The rest of the paper is organized as follows. We motivate the problem in Section 2. Our approach for addressing the problem is described in Section 3. Related work is discussed in Section 4. Conclusions are drawn and future work is discussed in Section 5.

2 Problem Motivation

Modern access control solutions predominantly follow the request-response paradigm, where the enforcement function (EF)—whether implemented in the security subsystem or in the application itself—intercepts each application access request, requests the decision from the decision function (DF), and enforces that decision provided in the response, as illustrated in Figure 1.

This separation into EF and DF enables DF re-use in the form of authorization servers, which leads to the lowering of the administrative overhead as well as the enforcement of consistent policies across multiple EFs (Figure 2). On the other hand, the request-response paradigm commonly leads to point-to-point (PTP) requests-responses

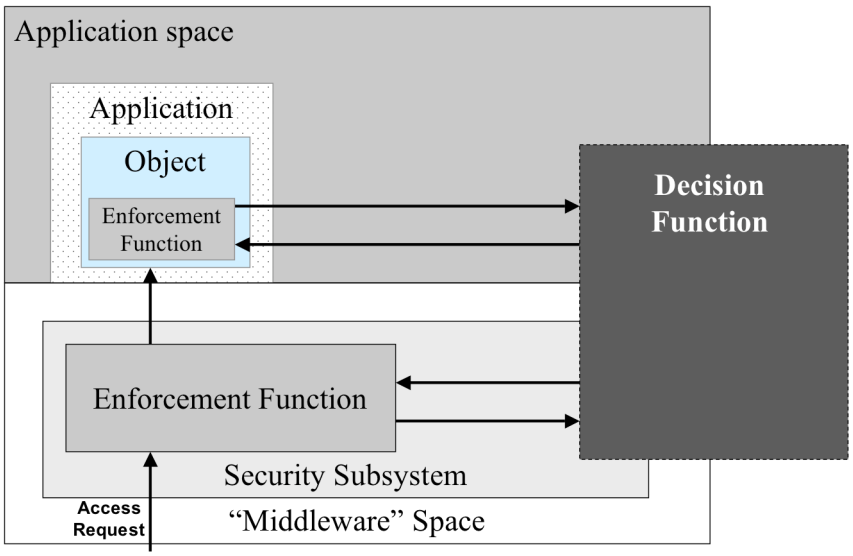

Figure 1. Request-Response Paradigm in Authorization Systems

with EFs obtaining decisions from DFs through synchronous remote procedure call (RPC). A number of modern access control systems follow the PTP approach. Representative examples are Access Manager [4], GetAccess [3], SiteMinder [6], and ClearTrust [12].

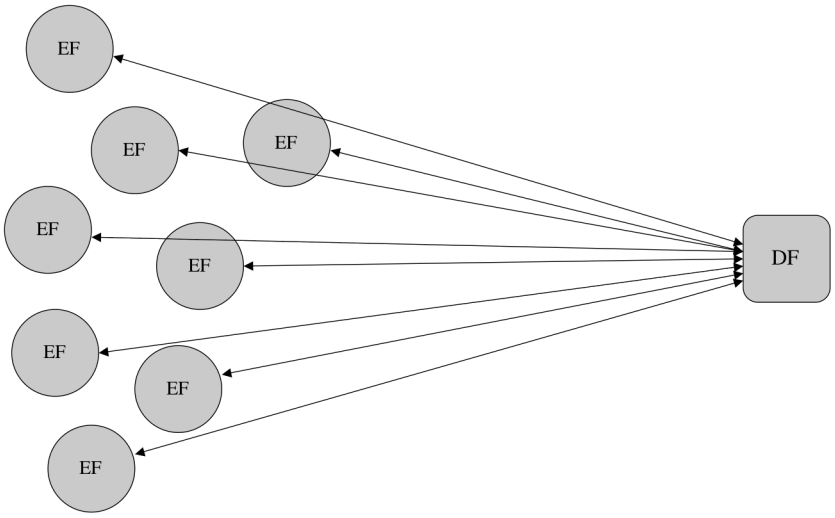

Figure 2. DF reuse for consistent enforcement and lower administrative overhead

The PTP approach breaks down at massive scale and/or with expensive to compute authorizations, resulting in fragile and sub-efficient solutions. Consider, for instance, an

enterprise with hundreds of thousands of networked commodity computers. Even if the mean time to failure (MTTF) of each computer (and all its critical software and hardware components) were one year, in an enterprise with half a million of computers,[1] over 1,300 would stop every day. With the average availability of every machine maintained at the level of 99.9% (somewhat unrealistically high expectation for such a large population of commodity equipment), 500 computers would be unavailable at any given moment.

In a large-scale enterprise, arriving to an authorization decision could be computationally expensive due to the heterogeneity and the large size of the object and subject populations. In addition, authorization policy evaluation could require obtaining various data from human resources, medical records, and other repositories of business data, which commonly increases the access control decision time further. The overall delay of requesting and obtaining an authorization could make the authorization overhead prohibitively expensive.

3 Proposed Approach

Our method is three-fold: the use of publish-subscribe approach for reducing the coupling between EFs and DFs, recycling authorizations published by DFs, and flooding delivery channels with speculatively computed authorizations.

3.1 Publish-Subscribe Architecture for Delivering Authorizations

First, we propose to utilize publish-subscribe architecture for requesting authorizations from DFs and delivering them to EFs, as illustrated in Figure 3. Implemented properly using message-oriented middleware (MOM) or some other mature technology, an authorization solution based on publish-subscribe architecture could result in the improvement of the resilience to frequent failures of the authorization servers and network partitioning frequently occurring in massive scale enterprises. With publish-subscribe, a request for an authorization could be delivered to many DFs. Even though

[1] Some enterprises, such as Amazon.com, are not far away from having that many computers with MTTF as short as two months [13].

some of them could fail, the chances that at least one will provide a response on time will be higher. In its turn, same authorization can be received by more than one EF enabling higher degree of authorization reuse.

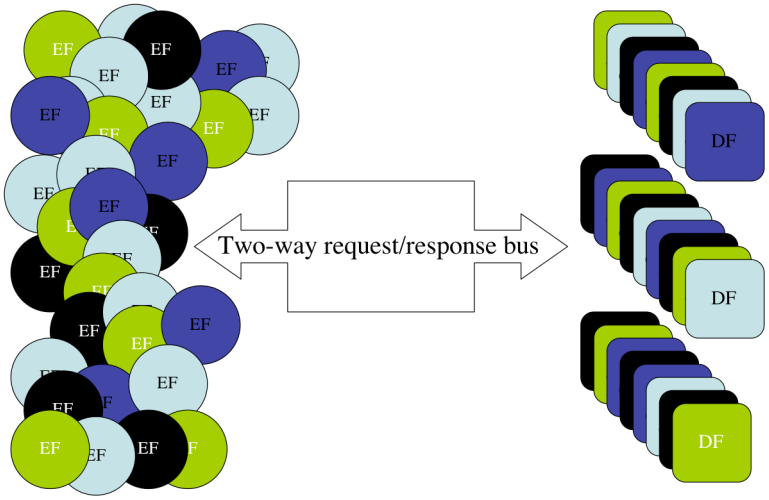

Figure 3. Publish-subscribe architecture for requesting and delivering authorizations

We are starting a technical feasibility study for the use of a publish-subscribe architecture for authorization infrastructures. During the study, we plan to get a better understanding of ways to balance resilience to DF failures with the minimization of redundant computations on the side of DFs. We also plan to compare an authorization system based on publish-subscribe architecture to a PTP-based one. To measure latency and other characteristics, we plan to use a modified version of Rice University Bidding System (RUBiS) benchmark modelled after eBay.com as a sample application based on PHP, Java Server Pages, and EJB.

Even with a publish-subscribe approach, the latency for requesting and obtaining an authorization is still bound by the computational cost of evaluating the credentials and the policy, as well as retrieving additional data used for policy evaluation, e.g., dynamic attributes in Resource Access Decision (RAD) architecture [2], or additional attributes of the protected resource. Furthermore, even mature publish-subscribe solutions may fail due to network partitioning, leaving EFs without authorizations to enforce. This is why the other two elements of our approach are important.

3.2 Sharing and Recycling Authorizations

The use of publish-subscribe approach also facilitates sharing and recycling of authorizations among those EFs whose populations of subjects and objects overlap. An authorization requested by one EF is received by all EFs subscribed to the same type[2] of authorizations, naturally leading to the re-use of that authorization. Furthermore, we suggested in [1] that not only "precise" but also "approximate" authorizations can be re-used.

In [1], we have also developed a preliminary version of Secondary and Approximate Authorizations Model (SAAM) that defines the notions of primary vs. secondary and precise vs. approximate authorizations. $SAAM_{SM}$, a variation of basic SAAM, also suggests a method of selecting the best approximate authorization for a given request from a set of existing authorizations. The method is applicable to a large class of authorization policies that satisfy two basic requirements: monotonic policies, and the abstraction of subjects as sets of name-value pair attributes. By adding another assumption, that an authorization policy could identify those subject attributes that were significant for arriving at the final authorization decision, the method for selecting the best approximate authorization is improved further.

3.3 Flooding EFs With Speculative Authorizations

Building on the publish-subscribe architecture and taking into account the extremely low cost of CPU resources and network bandwidth, we propose to speculatively pre-compute and publish authorizations. Like with junk mail, even if no EF needs every authorization produced by the DFs, those authorizations that turn out to be needed will be readily available with virtually no latency observed by EFs.

Similarly to speculative execution techniques used for processor architectures [8], we plan to develop methods for efficient predictions of which authorization request(s) should be computed next based on the history of prior requests. New studies are necessary to

---

[2] We do not define "authorization type" as the details of EF subscription have not been worked out yet.

determine whether the prediction algorithms should be application specific and which heuristics should be used. We expect a combination of static and dynamic speculation with the EF providing hints or maybe even pre-fetching eagerly authorizations for those resources that will be likely accessed by the subject in question. Alternatively, a DF can infer the type of the application from the authorization request and use that knowledge or the history of similar access sessions to predict future authorization requests.

4 Related Work

To the best of our knowledge, no prior work on the topic of employing publish-subscribe architectures for constructing authorization infrastructures, recycling or flooding authorizations has been published. The literature considers only the issues related to arriving at and enforcing access control decisions.

SAAM methods for recycling authorizations assume partial order dominance relation among subjects. Well-defined relations among objects, especially those in databases, could be also used to optimize authorization recycling further. A number of works exploit the specifics of the database objects relations to optimize authorization decision performance in a way similar to ours [5, 9, 10, 11]. SAAM however is not limited to database applications.

Due to the generality of our approach and little or no assumptions about the structure and format of authorization elements, it seems to be compatible with most traditional and modern access control models, including UCON [7].

To exploit more parallelism, processor designers have explored the idea of speculation, which allows the execution of an instruction before the processor know that the instruction should execute (i.e., it avoid control dependence stalls) [8]. The concept of speculation has its roots in the original 360/91 processor, which performed very limited form of speculation. There are two major approaches to speculation. The first is static speculation performed by the compiler with hardware support. In such schemes, the compiler chooses to make an instruction speculative and the hardware helps by making it easier to ignore the outcome of an incorrectly speculated instruction. Conditional

instructions can also be used to perform limited speculation. Speculation can also be done dynamically by the hardware using branch prediction to guide the speculation process.

5 Conclusions and Future Work

This paper suggested a new way of designing authorization solutions for massive-scale enterprises. Our approach exploits virtually free processor time and network bandwidth resources. It does not require highly specialized fault tolerance OSs or middleware, and, hopefully, allows decreasing the consumption of the most valuable resource in today IT infrastructures, human attention and time. The approach takes advantage of publish-subscribe architectures to share and recycle authorizations among instances of the security enforcement logic. In addition, we propose to speculatively pre-compute authorizations and flood the enforcement logic with them.

There are many questions to be answered before the validity and feasibility of our approach are understood, for example: How can several DFs collectively serve multiple EFs in an efficient manner? How efficient can algorithms be for finding suitable authorization(s) in a large stream and/or cache of already made authorizations? Can generic schemes for speculative pre-computing of authorizations suite most application domains or per-application customization will be necessary? How much can failure resiliency be increased with our approach comparatively to the traditional PTP-based?

At the time of writing this paper, a preliminary version of a model for recycling authorizations has been developed [1]. We are refining the model and starting a feasibility study of a publish-subscribe architecture for an authorization infrastructure. We also plan to start a project on developing schemes for speculative pre-computing authorizations.

References

[1] K. Beznosov, *Recycling Authorizations: Toward Secondary and Approximate Authorizations Model (SAAM)*, LERSSE-TR-2005-01, LERSSE, Dept. of Elec. and Comp. Engineering, University of British Columbia, March 2005.

[2] K. Beznosov, Y. Deng, B. Blakley, C. Burt, and J. Barkley, "A Resource Access Decision Service for CORBA-based Distributed Systems," in *Proceedings of Annual Computer Security Applications Conference*, Phoenix, Arizona, USA, 1999, IEEE Computer Society, pp. 310-319.

[3] Entrust, *getAccess Design and Administration Guide*, Entrust, September 20 1999.

[4] G. Karjoth, "Access control with IBM Tivoli Access Manager," *ACM Transactions on Information and Systems Security*, vol. 6, no. 2, 2003, pp. 232-57.

[5] R. Motro, "An Access Authorization Model for Relational Databases Based on Algebraic Manipulation of View Definitions," in *Proceedings of ICDE*, 1989, pp. 339-347.

[6] Netegrity, *SiteMinder Concepts Guide*, Netegrity, 2000.

[7] J. Park and R. Sandhu, "The UCONabc usage control model," *ACM Transactions on Information and System Security*, vol. 7, no. 1, 2004, pp. 128-174.

[8] D.A. Patterson and J.L. Hennessy, *Computer Architecture: A Quantative Approach*, 2nd ed., San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1996.

[9] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending Query Rewriting Technique for Fine-Grain Access Control," in *Proceedings of SIGMOD*, Paris, France, 2004.

[10] A. Rosenthal and E. Sciore, "View Security as the Basis for Data Warehouse Security," in *Proceedings of International Workshop on Design and Management of Data Warehouses*, 2000.

[11] A. Rosenthal and E. Sciore, "Administering Permissions for Distributed Data: Factoring and Automated Inference," in *Proceedings of 15th Annual Working Conference on Database and Application Security*, Niagara, Ontario, Canada, 2001, pp. 91-104.

[12] Securant, *Unified Access Management: A Model For Integrated Web Security*, Securant Technologies, June 25 1999.

[13] W. Vogels, *How Wrong Can You Be? Getting Lost on the Road to Massive Scalability*. Keynote speech at International Middleware Conference, Toronto, Canada, 2004.