Flooding and Recycling Authorizations

Konstantin (Kosta) Beznosov
Laboratory for Education and Research in Secure Systems Engineering*
Department of Electrical and Computer Engineering
University of British Columbia
2332 Main Mall
Vancouver, Canada
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 authorization servers through remote procedure calls. In massive-scale and complex enterprises, PTP authorization architectures result in fragile and inefficient solutions. They also fail to exploit virtually free CPU resources and network bandwidth. This paper proposes leveraging publish-subscribe architectures for increased reliability and efficiency by flooding delivery channels with speculatively pre-computed authorizations and actively recycling them on a just-in-time basis.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; C.2.0 [**Computer Communication Networks**]: Security and Protection; D.4.6 [**Security and Protection**]: Access Controls; F.1.2 [**Modes of Computation**]: Parallelism and concurrency

General Terms

Security

Keywords

authorization recycling, authorization flooding, access control, authorization, publish-subscribe, junk authorizations

1. INTRODUCTION

Our approach is based on four observations. First, microprocessor technologies have evolved to the point where processor resources are virtually free. Second, componentization of personal computers has driven the market to the point where commodity computing has become the most cost-effective approach. Specialized computer architectures, operating systems, and solutions have been driven out of the market, with large organizations steadily moving towards commodity operating systems (OSs) (e.g., Windows and Linux) running on farms of personal computers (PCs). Third, network bandwidth has followed the same pattern as CPU resources, albeit at a slower pace. Fourth, the cost of human time and attention continues to increase.

Given this context, we propose a novel approach to authorization architectures consisting of three key elements. First, we suggest decoupling the enforcement and decision parts of access control solutions with a publish-subscribe architecture. The administrative and operating overheads associated with reconfiguring access control systems to accommodate component and infrastructure failures should thereby be reduced. Second, since multiple policy enforcement points (PEPs) can subscribe to and share the same authorizations, we put forward a way of "actively recycling" authorizations by searching through the authorization streams and caches and finding the "best approximate" (as opposed to "precise") authorizations for the request to be authorized. Finally, we consider taking advantage of virtually free CPU resources and bandwidth by speculatively pre-computing authorizations and flooding the publish-subscribe channels with them.

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

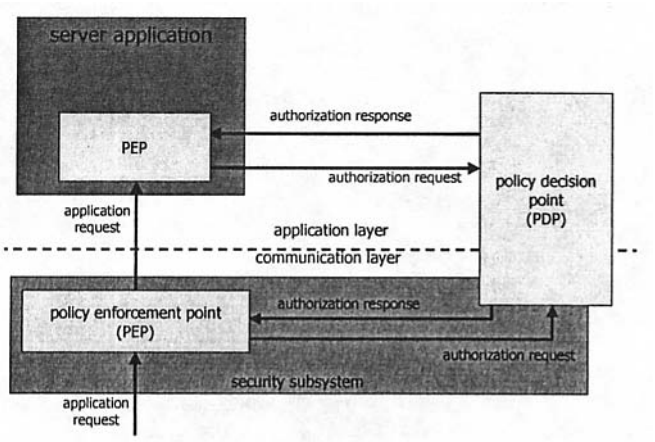
The main contribution of this paper is to suggest the three related approaches of publish-subscribe, active recycling of authorizations, and flooding PEPs with speculatively pre-computed "junk" authorizations as a promising alternative to current authorization infrastructures, which are based on remote procedure call (RPC) semantics and point-to-point architectures. While this paper is limited to only proposing these approaches, their validation is the subject of the ongoing research at Laboratory for Education and Research in Secure Systems Engineering (LERSSE).

The rest of the paper is organized as follows. We motivate the problem in Section 2, and address it in Section 3. We

*lersse.ece.ubc.ca

NSPW 2005 Lake Arrowhead CA USA

Figure 1: Request-response paradigm in authorization systems



discuss related work in Section 4, then draw conclusions and outline future work in Section 5.

2. PROBLEM MOTIVATION

Modern access control solutions mostly follow the request-response paradigm. In this paradigm, the policy enforcement point (PEP)—whether implemented in the security subsystem or in the application itself—intercepts each application access request, requests a decision from the policy decision point (PDP), and enforces that decision, as illustrated in Figure 1. The figure depicts two PEPs, illustrating two common locations for the enforcement logic to occur: either in the communication security subsystem or in the application itself. Depending on the nature of the communication and application, either one or both points in the request path are used for enforcing authorization and other security policies. The PDP is shown to span communication and application spaces, as it can be located in either space, also subject to communication and application architectures.

This separation into PEP and PDP enables reusing PDPs in the form of authorization servers, thereby reducing administrative overhead, as there are fewer PDPs to administer than PEPs,[1] as well as enforcing consistent policies across multiple PEPs (Figure 2). On the other hand, the request-response paradigm commonly leads to point-to-point (PTP) request-responses, with PEPs obtaining decisions from PDPs through synchronous RPC. A number of modern access control systems follow the PTP approach. Representative examples are Access Manager [7], GetAccess [5], SiteMinder [11], and ClearTrust [17]. The PTP approach breaks down at massive-scale or with expensive to compute authorizations, resulting in fragile and inefficient

[1]Although feasible in most enterprise applications, centralizing security administration is difficult to achieve in some applications—e.g., collaborative systems—where self-administration is the norm. Thanks to Mary Ellen Zurko for identifying this exception during discussion of the paper at the NSPW.

Figure 2: PDP reuse for consistent enforcement and lower administrative overhead

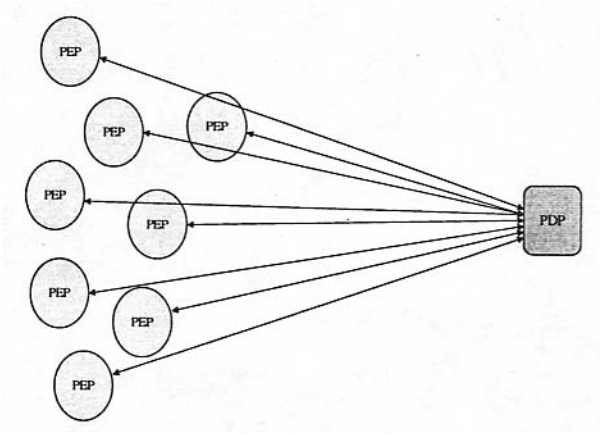
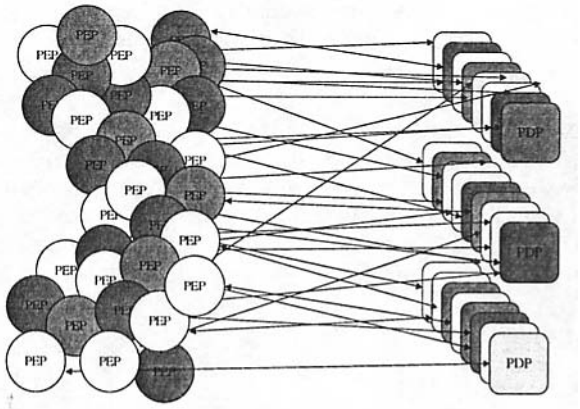


Figure 3: The high degree of coupling between PEPs and PDPs leads to fragile architectures

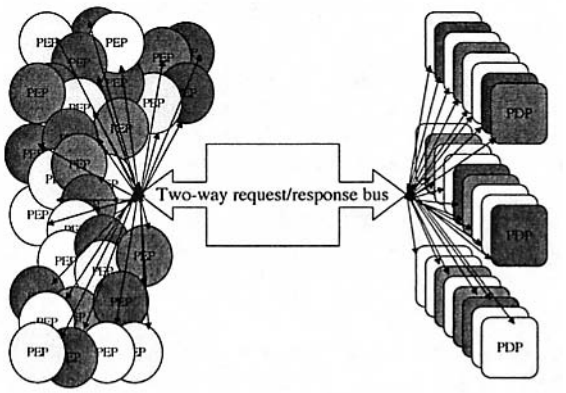


solutions. Figure 3 illustrates the high coupling of solutions based on PTP architectures, which, as a result, become also prone to high fragility. Consider, for instance, an enterprise with hundreds of thousands of networked commodity computers. Even if the mean time to failure (MTTF) of each computer (and all its critical software and hardware components) were one year, in an enterprise with half a million computers,[2] over 1,300 would fail every day. With the average availability of each machine maintained at a level of 99.9% (a somewhat unrealistically high expectation for such a large population of commodity equipment), almost 500 computers would be unavailable at any given moment.

In a large-scale enterprise, arriving at an authorization decision could be computationally expensive due to the heterogeneity and large size of the object and subject populations. In addition, authorization policy evaluation could require

[2]Some enterprises, such as Amazon.com, are not far away from having that many computers with MTTF as short as two months [19].

Figure 4: Publish-subscribe architecture for requesting and delivering authorizations



obtaining just-in-time data from human resources, medical records, and other repositories of business data, which commonly further increases the access control decision time. The overall delay in requesting and obtaining an authorization could make the authorization overhead prohibitively expensive.

3. PROPOSED APPROACH

Our approach is three-fold: the use of a publish-subscribe approach for reducing coupling between PEPs and PDPs; actively recycling authorizations published by PDPs; and flooding delivery channels with speculatively computed "junk" authorizations.

Although each of these techniques could be applied independently from the others, a publish-subscribe bus provides a good vehicle for the other two. For example, authorization recycling could be employed for PTP architectures, and speculatively computed authorizations could be delivered to PEPs using call-back mechanisms.

3.1 Publish-Subscribe Architecture for Delivering Authorizations

We propose using publish-subscribe architecture for requesting authorizations from PDPs and delivering them to PEPs, as illustrated in Figure 4. Implemented properly, using message-oriented middleware (MOM) or some other mature technology, an authorization solution based on publish-subscribe architecture could improve the resilience to frequent fail-stop failures[3] of authorization servers and network partitioning in massive-scale enterprises. With publish-subscribe, a request for an authorization could be delivered to many PDPs. Even though some of them could fail, the chances that at least one will provide a response on time will be higher. In turn, the same authorization can be received by more than one PEP, enabling a higher degree of authorization reuse.

Although it remains to be shown, we expect that replacing the coupling between PDPs and PEPs with couplings between PDPs and publish-subscribe channels, as well as between PEPs and those channels, could reduce the human resources required to operate and administer authorization infrastructures. Consider a failed PDP, for instance. It still needs to be brought back up and possibly relocated. However, only the configuration of the publish-subscribe infrastructure has to be re-adjusted and not all the PEPs that depend on the failed PDP.

We are undertaking a feasibility study of the use of publish-subscribe architecture in authorization infrastructures. We hope to gain a better understanding of ways of balancing resilience to PDP failures with minimizing redundant computations on the side of PDPs. We also plan to compare an authorization system based on publish-subscribe architecture with one based on PTP.

The use of publish-subscribe architecture has its own problems. The major one is that the publish-subcribe bus becomes a single point of failure for the whole authorization infrastructure, as its failure renders all PDPs unavailable to any PEP connected to them through the bus. Another concern is the fitness of a particular publish-susbcribe implementation for handling traffic between PEPs and PDPs without becoming a bottleneck. The ways of addressing these concerns are specific to particular publish-subscribe architecture and are beyond the scope of this work.

Even with a publish-subscribe approach, the latency for requesting and obtaining an authorization is still bound by the computational cost of evaluating credentials and policy, and of 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 PEPs without authorizations computed by PDPs. For these reasons, the other two elements of our approach are also important.

3.2 Sharing and Actively Recycling Authorizations

The publish-subscribe approach also facilitates sharing and recycling of authorizations among those PEPs whose populations of subjects and objects overlap. An authorization requested by one PEP is received by all PEPs subscribed to the same type[4] of authorizations, leading to re-use of that authorization. Both "precise" and "approximate" authorizations can be re-used, as we suggested in [1].

[3]The fail-stop failure model makes two assumptions about the failure behaviour of processes: processes fail only by permanently crashing, and when a process crashes, surviving processes will eventually detect that failure. This work does not attempt to address general, Byzantine, failures in which failed processes keep communicating by sending possibly wrong data. In other words, we assume that every PDP either responds with correct authorizations or does not respond at all.

[4]We do not define "authorization type", as we have not yet worked out the details of PEP subscription. One possible candidate for grouping authorizations into types is by subject and object populations. That is, PEPs would receive the same authorizations if they serve overlapping populations of subjects and objects.

In [1], we also developed a preliminary version of a Secondary and Approximate Authorizations Model (SAAM) that defines the notions of primary vs. secondary and precise vs. approximate authorizations. Most importantly, SAAM suggests algorithms for deriving new authorizations using past ones. $SAAM_{SM}$, a variation of basic SAAM, 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: that they are monotonic, and that subjects can be abstracted as sets of attributes, each having a name and a value. By adding another assumption, that an authorization policy could identify subject attributes that were significant for arriving at the final authorization decision, the method for selecting the best approximate authorization should be further improved.

Publish-subscribe channels are commonly implemented with store-and-forward schemes. We therefore anticipate that authorizations currently in a publish-subscribe channel can be searched using techniques similar to those used for searching in a static cache. The applicability of similar search techniques makes the stream of authorizations coming through publish-subscribe channels similar to an authorization cache whose content is ever changing.

So far our work on SAAM has focused on infrequently changing policies where authorizations computed by the PDP are mostly independent of time. Even though we have not yet developed recycled authorization consistency mechanisms for policies that undergo frequent changes, it is expected that conventional techniques and protocols for cache consistency (such as those used in HTTP response caching [6]) will be adequate. We regard the problem of time-of-check-to-time-of-use as a particular case of the cache consistency problem. We expect that the former should be resolved in addressing the latter, and therefore see no need to develop a specialized solution such as introducing a short delay in the delivery of an authorization to allow PEP cache content to quiesce.

However, history-sensitive policies (e.g., Chinese Wall [3], High Watermark model [20]), dynamic separation of duties [10], or other types of policies that require subject rights to be consumable (e.g., task-based authorizations [18]) cannot be supported simply through enforcing cache consistency. Our approach is to develop SAAM algorithms tailored to each access control model. To support the Chinese Wall model, for instance, it will be necessary to develop corresponding SAAM algorithms.

3.3 Flooding PEPs with Speculative Authorizations

Building on 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 "junk" authorizations. As with junk mail, even if no PEP needs every authorization it receives, those authorizations that are needed will be readily available with virtually no latency observed by PEPs, ultimately improving the end-user's experience. This is where the borderline between PEP and PDP begins to blur, as PDPs effectively "push" the policy, encoded in the form of authorizations, to the PEPs [4]. As was also observed by the NSPW participants, in the extreme case the PEP *becomes* the PDP (similar to a capability-based system). Understanding the implications of pushing policies from the PDP to its PEPs in the form of pre-computed authorizations is the subject of future research.

We plan to develop methods for efficient predictions of which authorization request(s) should be computed next, based on the history of prior requests. These methods will be similar to speculative execution techniques used for processor architectures [13]. New studies are necessary to determine whether the prediction algorithms should be application specific and which heuristics should be used.

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 speculatively pre-computed authorizations has been published. The literature considers only the issues related to arriving at and enforcing access control decisions.

Current SAAM methods of recycling authorizations assume a partial order dominance relation among subjects. Well-defined relations among objects, especially those in databases, could also be used to further optimize authorization recycling. A number of reported studies exploit the specifics of database object relations to optimize performance of authorization decisions in a way that is similar to ours [9, 14, 15, 16]. SAAM, however, is not limited to database applications. The generality of our approach and its minimal or no assumptions about the structure and format of authorization elements, makes it compatible with most traditional and modern access control models, including UCON [12].

To exploit more parallelism, processor designers have explored the idea of speculation, which allows the execution of an instruction before the processor "knows" that the instruction should execute (i.e., it avoids control-dependence stalls) [13]. The concept of speculation has its roots in the original 360/91 processor, which performed a very limited form of speculation. There are two major approaches to speculation in processor architectures. 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. We expect a combination of static and dynamic speculation in our architecture, with the PEP providing hints or maybe even eagerly pre-fetching authorizations for those resources to be accessed likely by the subject in question. Alternatively, a PDP can infer the type of application from the authorization request and use that knowledge or the history of similar access sessions to predict future authorization requests.

5. DISCUSSION AND CURRENT STATUS

This paper suggests a new way of designing authorization solutions for massive-scale enterprises. Our approach exploits virtually free processor time and network bandwidth. It does not require highly specialized, fault-tolerant OSs or middleware, and, hopefully, it will reduce the consumption of the most valuable resources in today's IT infrastructures, human attention and time. The approach takes advantage of publish-subscribe architectures to share and actively recycle authorizations among instances of security enforcement logic. In addition, we propose speculatively pre-computing "junk" authorizations and flooding the enforcement logic with them to further reduce latency associated with computing authorizations and communicating them to PEPs.

There are many questions to be answered before the validity and feasibility of our approach are established. For example, how can several PDPs collectively serve multiple PEPs in an efficient manner? How efficient can the algorithms for finding suitable authorization(s) be in a large stream and/or cache of already-made authorizations? Can generic schemes for speculatively pre-computing authorizations suit most application domains or will customization by application be necessary? How much can our approach improve reliability of a PEP in the case of PDP failure, compared to traditional approaches? With authorizations flooding publish-subscribe channels and with the lower degree of coupling between PDPs and PEPs, how can a sufficient level of trust in the published authorizations be maintained? Will cryptographic protection of the authenticity and integrity of the access control decisions be feasible and, if so, sufficient for maintaining the trust? If authorizations need to be confidential and not revealed to the rest of the world, how can they be made readily available to thousands of PEPs, which "come and go"? How likely is the proposed approach to be effective in environments where authorization policy changes frequently or is time-dependant, or where authorizations depend on access history, as in Chinese Wall [3] policies? How can policies based on *consumable rights*, or more generally *mutable attributes* [12], of subjects be supported without making PEP cache "stateful"? Does a PDP need to "remember" past speculations to be able to speculate accurately in the future?

At the time of writing, a preliminary version of a model for recycling authorizations is being developed [8]. We are refining the model and developing a simulation-based study of our key claims: improvements of PEP's resiliency to PDP's (or the connecting network's) fail-stop failures, as well as improved latency due to saved round-trips between PEPs and PDPs. We are also developing an architecture for sharing recycled authorizations among PEPs via distributed hash tables, which are common in peer-to-peer networks.

Acknowledgment

The author is grateful to the anonymous reviewers of this paper as well as to the participants of NSPW '05, who provided insightful feedback on earlier versions. Jason Crampton and Wing Leung gave helpful comments on the pre-proceedings version of the paper.

6. 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, IEEE Computer Society, pp. 310–319, 1999.

[3] Brewer, D., and Nash, M. "The Chinese Wall security policy," in *Proceedings of the IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 206–214, May 1989.

[4] J. Crampton (private communication), 2005.

[5] Entrust Inc., *GetAccess Design and Administration Guide*, September 20, 1999.

[6] James Gwertzman and Margo I. Seltzer, "World wide web cache consistency," in *USENIX Annual Technical Conference*, pages 141–152, 1996.

[7] G. Karjoth, "Access control with IBM Tivoli Access Manager," *ACM Transactions on Information and Systems Security*, vol. 6, no. 2, 2003, pp. 232–257.

[8] W. Leung and J. Crampton and K. Beznosov, *Toward Secondary and Approximate Authorizations Model (SAAM)*, technical report, LERSSE, Dept. of Elec. and Comp. Engineering, University of British Columbia, in progress.

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

[10] M. Nash and L. Poland. "Some conundrums concerning separation of duty," in *Proceedings of the Symposium on Security and Privacy*, (Oakland, CA, May 1990), IEEE Computer Society Press, pp. 201–207.

[11] Netegrity Inc., *SiteMinder Concepts Guide*, 2000.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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 Falls, Ontario, Canada, 2001, pp. 91–104.

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

[18] R. Thomas and R. Sandhu, "Task-based Authorization Controls (TBAC): Models for Active and Enterprise-Oriented Authorization," In *Management Database Security XI: Status and Prospects*, North-Holland, Amsterdam, 1997.

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

[20] C. Weissman. Security controls in the ADEPT-50 timesharing system. In *AFIPS Conference Proceedings*, v. 35, pp. 119–133. FJCC, 1969.