XACML Policy Performance Evaluation Using a Flexible Load Testing Framework

Bernard Butler, Brendan Jennings, Dmitri Botvich
FAME
Telecommunications Software & Systems Group
Waterford Institute of Technology
Ireland
{bbutler,bjennings,dbotvich}@tssg.org

ABSTRACT

The performance and scalability of access control systems is growing more important as organisations deploy ever more complex communications and content management systems. Fine-grained access control is becoming more pervasive, so decisions are more frequent and policy sets are larger. We outline a flexible performance testing framework that accepts XACML PDP implementations (in the server component) and submits representative access control requests (from the client component) in a representative temporal ordering. The framework includes instrumentation and analysis modules to support performance experiments. We describe an initial realization of the framework and report on initial experiments comparing the performance of the SunXACML and Enterprise XACML PDPs.

Categories and Subject Descriptors

D.2.8 [**SOFTWARE ENGINEERING**]: Metrics—*Performance measures*; D.4.6 [**OPERATING SYSTEMS**]: Security and protection—*Access controls, Information flow controls*

General Terms

Security, Performance, Measurement

Keywords

Access control policies, performance evaluation, measurement testbed

1. INTRODUCTION

Policy Decision Point (PDP) *performance* is an important access control system requirement. In larger organisations, access decisions depend on the context of an access request, so *fine-grained* access control is needed to implement security policies with complex boundaries between permitted and denied behaviour. There are more access requests, hence policy evaluations and each policy evaluation takes longer as policy sets grow larger.

As an example, policy control of instant messaging communications in enterprises causes large numbers of policy evaluations, particularly in group-chat scenarios, where the access control system must decide which participant pairs can communicate. Such policy control is needed in organisations where Chinese Walls [1] must be maintained between groups for regulatory reasons.

Many enterprise-level access control systems encode access controls as XACML (the eXtensible Access Control modelling Language) [7] hence researchers focus on XACML policies and requests and their use in PEPs (Policy Execution Points) and XACML-based PDPs.

It is relatively easy to scale out the (stateless) PEP function, but not the (stateful) PDP function. Typical performance measures of a PDP set include *latency* and *throughput*, so any testbed needs to compute both. Some researchers advocate "black box" approaches such as caching frequently encountered request-result pairs. Alternatively, given one or more one of the policy set, request profiles or PDP source code, "white-box" approaches are possible. XACML policies can be *improved* by categorisation, reordering and clustering [4], numericalisation and simplification to tree structures [3], etc. XACML policies can also be *replaced* with an equivalent Description Logic formulation [2].

Generally the evidence presented by researchers is based on comparisons with the Sun XACML reference implementation [8] often using unpublished policies and requests. Hence it is difficult to compare one approach with another, or to determine what tradeoffs occur. We propose a *performance testbed* for access control implementations to facilitate research into the performance and scalability problems facing XACML-based access control. The aim of our work is to provide a *flexible* (easily configured) framework, enabling researchers to perform quantitative *experiments* under representative, controlled and repeatable conditions.

2. RELATED WORK

The problem of generating a large and representative set of policy requests for performance evaluation is related to that of generating a test set that covers as many of the policy conditions as possible. By ensuring full coverage, *all* policy conditions are checked and so there is a path to each terminal node in the decision tree inferred from the policy set [5]. [5] also describes how Margrave can be used to determine redundant rules in a complex policy set. [6] describe how policy mutation testing may be used to determine how well a given test set of XACML requests discovers faults (deliberately injected as *mutations*) in policy sets.

Data clustering has been applied to characterise policies and hence improve PDP performance [4].

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.
ACM 978-1-4503-0244-9/10/10.

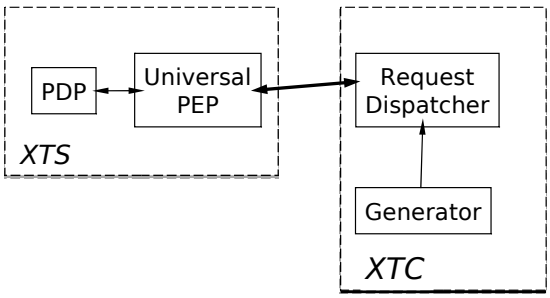


Figure 1: XACML Load Testing System Architecture.

3. FRAMEWORK OVERVIEW

The architecture of our XACML load testing framework is presented in Figure 1. The `XTS` (server) comprises a PDP and a simplified "universal" PEP and specific PDP *adapter*. Each PDP implementation needs an adapter to wrap calls from the universal PEP. The adapter also brackets each PDP call with timing calls to compute the elapsed time *at the PDP*.

Several modes of request generation are possible in the `XTC` (client)

- Reusing existing requests by weighted resampling
- Generating new requests, motivated by Section 2

`XTC` submits the generated XACML requests to the `XTS` PEP based on *request scenarios* chosen by testbed users. `XTA` persists the `XTS` results (both responses and timings) and analyses it offline by fitting empirical (frequency) distribution functions (*edf*) and clustering requests by locating peaks in the response duration *edf*.

4. INITIAL EXPERIMENTS AND RESULTS

In the testbed, we compared the performance of the Sun XACML and Enterprise XACML [9] PDP implementations, using the same policies and identical resampled requests, see Figure 2. Note that both PDPs made the same access decisions for all requests. We believe each cluster of processing durations (equivalently, *edf* peak) corresponds to a set of requests that share similar processing requirements such as policy search paths for a given PDP. Qualitatively, the Enterprise XACML PDP has better performance ($\bar{t}_{\mathrm{EX}} < \bar{t}_{\mathrm{SX}}$) than the Sun XACML PDP. In fact, the Enterprise XACML PDP shows remarkably consistent performance at about 1.4 milliseconds processing time per request, with just a few upper outliers.

The experiments are valid *for that combination of policies and requests* since we employ a randomised block design and control for other known factors.

5. CONCLUSIONS AND FUTURE WORK

The testing framework provides necessary infrastructure for building access control policy evaluation models. Such models are needed to understand how to improve performance and/or scalability of a XACML-based access control system. The framework can be used to test hypotheses regarding how to improve access control performance and scalability. For the future, there will be better request generation and timing analysis techniques, to broaden the scope of the hypotheses that can be tested. We can extend to more PDP implementations and multiple PDP instances. We will also look for a predictive model underlying the observed timings.

6. ACKNOWLEDGMENTS

The authors acknowledge the contribution of Keith Griffin and Ger Lawlor, Cisco Systems Inc., who helped clarify the requirements for modelling the performance of XACML PDPs. The work was funded by Science Foundation Ireland via the "FAME" Strategic Research Cluster, grant no. 08/SRC/I1403.

7. REFERENCES

[1] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[2] V. Kolovski, J. Hendler, and B. Parsia. Analyzing web access control policies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 677–686, New York, NY, USA, 2007. ACM.

[3] A. X. Liu, F. Chen, J. Hwang, and T. Xie. Xengine: a fast and scalable XACML policy evaluation engine. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 265–276, New York, NY, USA, 2008. ACM.

[4] S. Marouf, M. Shehab, A. Squicciarini, and S. Sundareswaran. Statistics & Clustering Based Framework for Efficient XACML Policy Evaluation. In *POLICY '09: Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks*, pages 118–125, Washington, DC, USA, 2009. IEEE Computer Society.

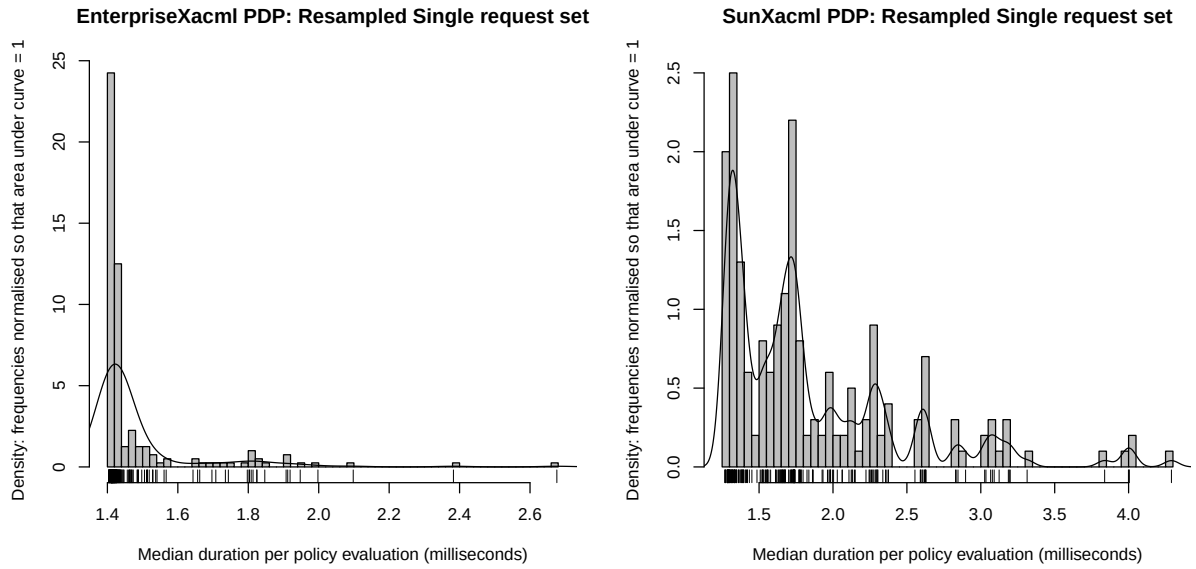
[5] E. Martin. Automated test generation for access control policies. In *OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 752–753, New York, NY, USA, 2006. ACM.

[6] E. Martin, T. Xie, and T. Yu. Defining and measuring policy coverage in testing access control policies. In *Proc. 8th International Conference on Information and Communications Security*, pages 139–158, 2006.

[7] T. Moses. eXtensible Access Control Markup Language TC v2.0 (XACML), February 2005.

[8] S. Proctor. Sun's XACML Implementation - Programmer's Guide for Version 1.2. http://sunxacml.sourceforge.net/guide.html, July 2004. Last accessed 2009-11-25.

[9] Z. Wang. Enterprise Java XACML. http://code.google.com/p/enterprise-java-xacml/wiki/DevelopmentPlan, February 2010. Last accessed 2010-04-19.



(a) Enterprise XACML PDP evaluation duration frequencies.

(b) Sun XACML PDP evaluation duration frequencies.

Figure 2: Comparison of the performance profiles of two XACML PDPs on the same policy and request sets.