Study of Secondary and Approximate Authorizations Model (SAAM)

Kyle Zeeuwen
kylez@ece.ubc.ca

Abstract— **Request response access control systems with off-site Policy Decision Points have their reliability and latency bounded by the network connecting the components. We demonstrate that recycling pre-computed authorizations based on knowledge of the BLP protocol reduces the latency and increases the reliability of the system to a greater extent than existing approaches that use precise authorization recycling.**

A simulation is described that measures latency and system reliability for systems that use no recycling, precise recycling, and approximate recycling.

Systems with a round-trip time around 20ms to their Policy Decision Points can experience an order of magnitude reduction in their average client latency by using SAAM Recycling.

I. Introduction

Request-response access control systems separate application logic from access decisions. Access is granted or denied based on the security policy of the system. The policy decision point (PDP) is also separated from the policy enforcement point (PEP) in many systems. This design is illustrated in Figure 1. Examples of this type of system include Access Manager [1], GetAccess [2], SiteMinder [3], and ClearTrust [4]. This design allows multiple PEP's to use a single PDP and reduces the administrative overhead placed on IT personnel. This design imposes an upper limit on the reliability of the system: if the PDP or the network becomes partitioned fails the system will also invariably fail. The separation of PDP from PEP also increases the latency observed by the users of the system. [5]

One improvement to this design is to reuse previously computed (secondary) authorizations. Existing approaches recycle an authorization if it exactly matches the pending authorization request [6]. This is referred to as precise authorization recycling. Another method, referred to as Secondary and Approximate Authorizations Model (SAAM), combines knowledge of the security model with non-precise, or approximate, secondary authorizations to determine the result of pending authorization requests [7].

It is believed that a recycling component that leverages approximate authorizations will reduce the average latency of processing authorization requests when compared to a similar system using precise recycling. The SAAM recycling component will also increase the reliability of the system by providing an alternative source of authorizations; the PDP will have to fail and the recycling component will have to miss for the system to fail.

This document presents results from ongoing research that uses a SAAM recycling component in a request-response access control system based on the Bell LaPadula (BLP) Mandatory Access Control policy [8], [9].

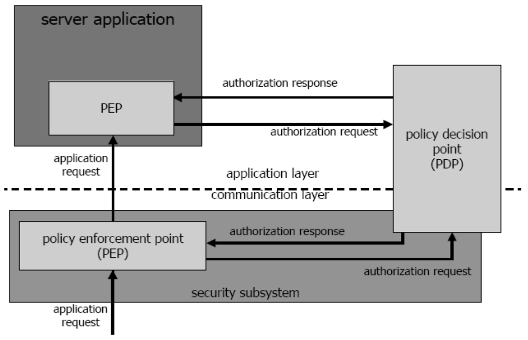

Fig. 1. Request-response paradigm in authorization systems

Results indicate that a SAAM recycling component produces a cache hit rate that is greater than a precise recycling component under the same circumstances. This increased hit rate translates into greater improvements in terms of latency and reliability. The extent of these improvements is presently under investigation.

The rest of the paper is organized as follows. Section II will introduce the components of the

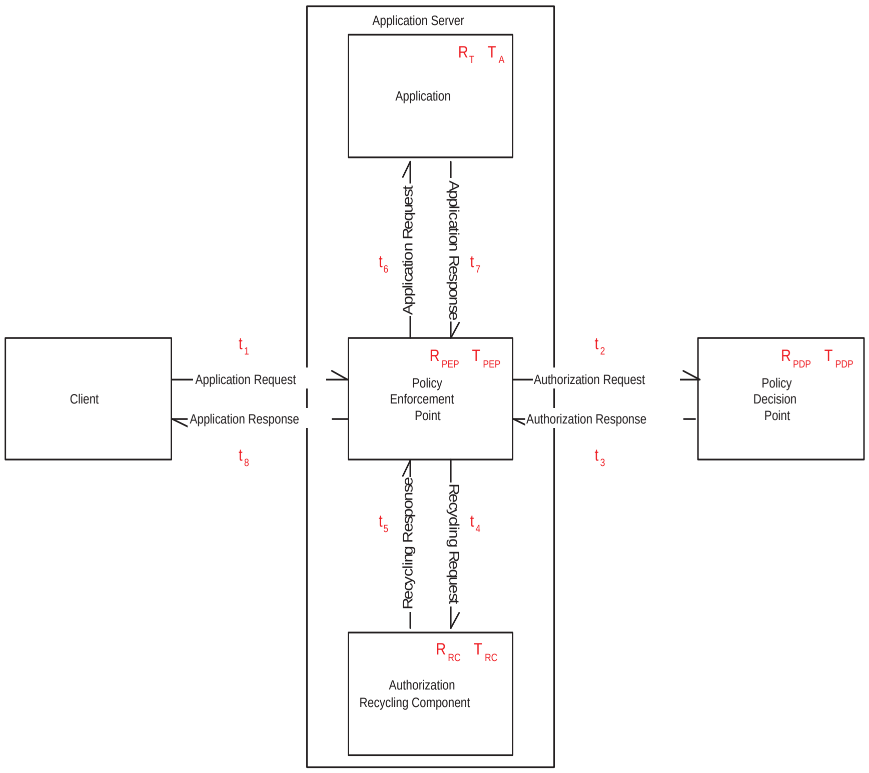

Fig. 2. Request-Response Access Control System with Recycling Component

simulation and provide formulas used to calculate latency and reliability. Section III presents and discusses the results that were gathered, and finally Section IV concludes.

II. Methodology

The methodology section presents an analytical model of the system and discusses which parts are important for our study and which can be safely ignored. Once this is complete the design and implementation of the simulation are presented.

A. Analytical Model of System

The analytical model involves interactions between 5 components, as shown in Figure 2. The client component represents one or more machines running client side software that issue requests to the system. The Policy Enforcement Point (PEP) intercepts these requests and generates authorization requests. These authorization requests are simultaneously sent to the Policy Decision Point (PDP) and the Recycling Component (RC). The PEP will use the first authorization response it receives to either allow or deny access to the requested resource. The resource resides in the application component. The PEP is also responsible for updating the RC with each request-response pair generated by the PDP.

The PDP contains all the BLP specific security information for the subjects and objects of the system. When it receives a request, the PDP determines if access should be allowed or denied. It communicates this decision back to the PEP via an authorization response.

The RC receives authorization requests from the PEP and searches for suitable matches. If an existing valid authorization request is found then the RC returns the corresponding authorization response to the PEP. Otherwise, the RC indicates that it cannot find a match. If the RC is using precise recycling only then it will simply search it's cache for a request that matches the pending request exactly. If the RC is using precise and approximate recycling it issues simultaneous requests to both components and uses the first response it receives. The details of the BLP approximate recycling algorithm are presented in [10].

1) Simplifying Assumptions: Several components of the analytical model can be safely ignored for the purposes of our simulation. Simplifying the simulation serves several purposes. First, the implementation becomes easier. Second, the correctness of the design is easier to verify. Lastly, the correlation between results and input becomes more pronounced as the number of inputs decreases.

The behavior of the Application is not important because the simulation is only concerned with measuring the effects of SAAM recycling. The only significant attributes of the application which must be considered are its contribution to the latency and reliability observed by the client.

It is not necessary to treat the Client as a unique entity. A list of requests can be substituted in place of the client. It is still necessary to account for the network latency between the client and the PEP.

Of all the timing parameters shown in Figure 2, only l_2, l_3, T_{PDD}, l_4, l_5, and T_{RC} affect the behavior of the simulation. The remaining values can be combined into a constant that affects the observed latency at the client.

The reliability of the application and the PEP can be combined into a constant that only affects the observed reliability of the client. Although it true that the reliability of the PEP affects the behavior of the PDP and RC, the purpose of this study is not to determine these effects. Including PEP failures into our simulation would unnecessarily complicate the analysis and interpretation of results.

2) Reliability Calculations: The reliability of the system can be calculated from the reliability of the components of the system. First we must construct a Reliability Block Diagram (RBD) [11] for the system. This allows us to visualize the failure paths of the system. The RBD is shown in Figure 3. The PEP represents a unique component because each request is handled by the PEP several times during processing. As a result it occurs several times in our RBD.

In typical software systems reliability is the probability that a system will function as intended at a given time. In our simulation we will treat the RC reliability in a unique manner. If the RC component returns a response when queried then it functioned 'as intended'. When the component fails to return a response we will consider this a failure, even though this is not a failure from a software engineering perspective. For the rest of this paper we will use the terms *reliability* and *hit-rate* interchangeably when referring to the RC.

The treatment of the RC's hit-rate as its reliability introduces another problem: the hit-rate does not vary directly with time like most standard software reliability measurements. Instead, the hit-rate is a function of the cache warmness. To accommodate this requirement the reliability of the system will be computed as a function of cache-warmness and time.

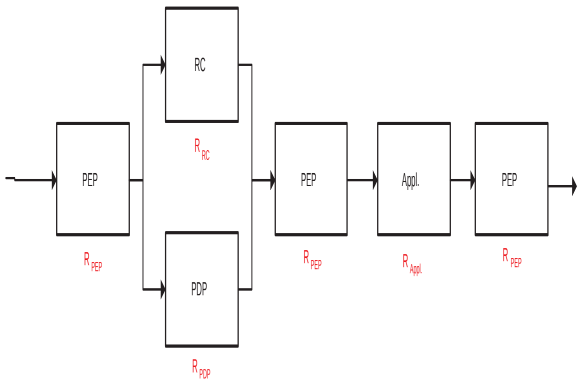

Fig. 3. Reliability Block Diagram for Simulation

Using standard reliability modeling techniques [11] we arrive at the following equation for system reliability.

$$\begin{aligned} R_{SYS}(t,x) = \quad & R_0(t) * (R_{PDP}(t) + R_{RC}(x) \\ & -R_{PDP}(t)R_{RC}(x)) \qquad (1) \\ R_0(t) = \quad & R^3_{PEP}(t) * R_{Appl}(t) \end{aligned}$$

We are concerned with determining the reliability improvement offered by using a SAAM component. This can be accomplished without modeling the reliability of the system over time. We can assume a fixed reliability for each time-dependant component and determine the effect of cache warmness on the reliability of the system. This allows us to further reduce Equation 1 to the following form:

$$\begin{aligned} R_{SYS}(x) = \quad & R_0 * (R_{PDP} + R_{RC}(x) - \\ & R_{PDP}R_{RC}(x)) \qquad (2) \end{aligned}$$

3) Latency Calculations: Latency will be measured from the time the client issues a request to the time it receives the response. This will be computed as a sum of the communication delays and the processing times for each of the components involved in processing the request.

The PEP uses the first authorization response it receives; therefore, there must be some logic used to determine which communication delays to use for each latency calculation. The communication costs for the PDP response time and for the RC response time are calculated and the lesser is used. Equation 3 shows the latency calculations. The terms are taken from Figure 2. The terms grouped to form L_0 will be treated as a single constant and not simulated. They can be varied during the analysis phase to determine their effect.

$$
\begin{aligned}
L = \quad & L_0 + min((t_2 + T_{PDP} + t_3), \\
& (t_4 + T_{RC} + t_5)) \qquad (3) \\
L_0 = \quad & t_1 + t_8 + t_6 + t_7 + T_{PEP} + T_A
\end{aligned}
$$

Equation 3 can be used to derive an equation for average latency based on the hit-rate of the recycling component. This is shown in Equation 4. This is done based on the observation that the latency is comprised of the PDP round trip time ($t_2 + T_{PDP} + t_3$) when the RC does not produce a response. In the case where the RC does produce a response, the minimum function from Equation 3 is invoked.

$$
\begin{aligned}
\overline{L} = \quad & L_0 + R_{RC}(x) * \\
& min((t_2 + T_{PDP} + t_3), (t_4 + T_{RC} + t_5)) \\
& +(1 - R_{RC}(x)) * (t_2 + T_{PDP} + t_3) \qquad (4)
\end{aligned}
$$

B. Simulation Design

The equations presented in II-A provide a way to calculate the reliability and latency observed at the client. The equations require the following values from the simulation: $T_{PDP}, T_{RC}(x)$, and $R_{RC}(x)$. A framework to produce these values is presented in this section. This framework is shown in Figure 4. Each component will be discussed.

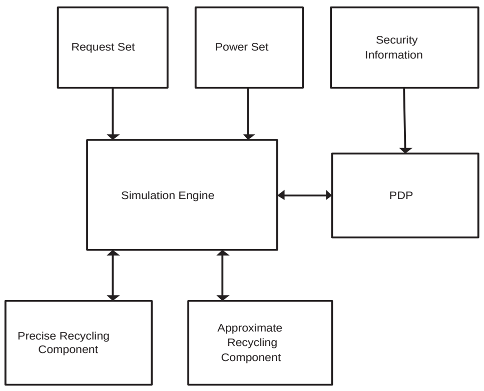

Fig. 4. Simulation Framework

1) Input Generation: The request set and power set are simply lists of requests. Each request is made up of a subject, object, and access type. The power set is a randomized list of every possible request given a set of subjects, objects, and access rights. The request set is also a randomized list of requests. The total size of the request set is an integer multiple of the size of the power set.

The security information contains the BLP specific security information for the subjects and objects in the system. It is generated randomly given the number of subjects, objects, classifications/clearances, and categories in the system.

2) PDP: The PDP implements the BLP protocol as specified in [8], [9].

3) Precise Recycling Component: The Precise Recycling Component (PRC) is implemented as a hash table. The requests serve as keys that map to one corresponding response. When the engine queries the PRC with a request the PRC performs a lookup using the request as a key. If a corresponding response is found then it is returned to the engine.

4) Approximate Recycling Component: The Approximate Recycling Component (ARC) leverages knowledge of the BLP protocol to infer the result of requests based on the results of previous request-response pairs. The details of this component can be found in [10].

5) Simulation Engine: The simulation engine is responsible for running the simulation and gathering the results. The engine reads requests from the power set and submits each to the PDP, the Precise Recycling Component (PRC), and the Approximate Recycling Component (ARC). The engine uses the responses from the PDP to update the PRC and ARC, which warms the cache to a specified level. Once a desired cache warmness is achieved the engine stops using requests from the power set and switches to the request set. It then iterates through each request in the request set and determines values for $T_{RC}, T_{PDP}, and R_{RC}$. During this phase the recycling components are not updated. This process repeats several times until the cache has been completely warmed. The result of this are sequences of $T_{RC}, T_{PDP}, and R_{RC}$ values which vary based on cache warmness.

III. Results and Discussion

Results are still being actively gathered and interpreted at the time of paper submission. This section presents some preliminary figures and details the ongoing work. The discussion of results is also presented in this section. Simulation results were gathered on a commodity PC using an Intel Pentium 4 2.8 GHz hyper-threaded processor with 1 GB of RAM. The OS on this machine was Windows XP Professional SP2. The simulation was written in Java and run on the Sun's 1.4.2_09 JRE. High resolution timing was accomplished using the Sun.Misc.Perf Java class.

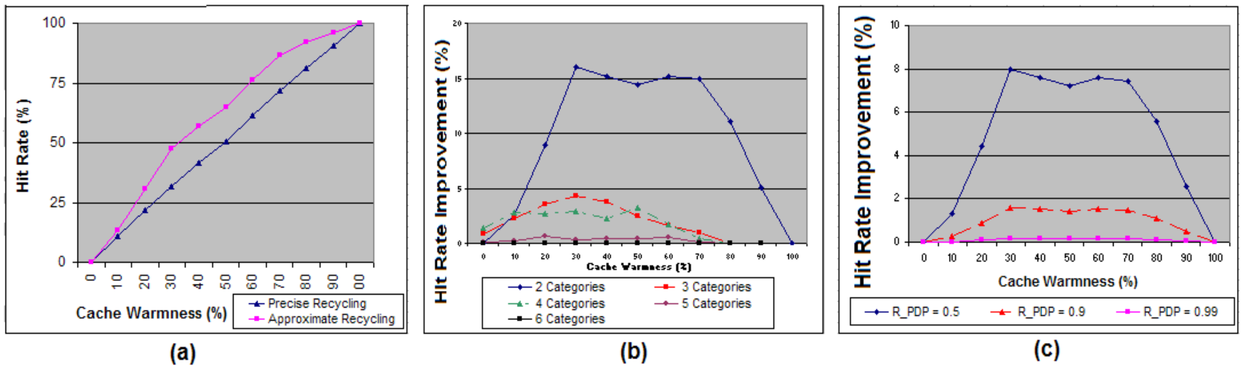

Fig. 5. Graph (a) compares the hit rate produced using SAAM Recycling and Precise Recycling. (b) and (c) plot the improvement of SAAM Recycling over the Precise Recycling. (b) Shows how the hit rate improvement diminishes as the the BLP lattice grows. (c) shows the difference in observed system reliability for different values of PDP reliability.

Three distinct architectures were simulated. The first uses no authorization recycling. The second uses only precise recycling, and the third uses both precise and SAAM recycling.

Unless otherwise indicated, the results presented were gathered using a data set comprised of 10 subjects and 20 objects. Subjects and objects were assigned levels in a small BLP security lattice consisting of 2 security clearances and 2 security categories. The power set consisted of 400 requests and the request set 1200 requests.

A. *Reliability Measurements*

Figure 5(a) shows the hit rate as a function of cache warmness for the data set described above. This data set is too small to show the be effects of SAAM Recycling in a real scenario; however, it does give evidence that the SAAM recycling component produces a higher hit rate than a precise component for an identical series of requests.

The same data is used as input to Equation 2 to produce Figure 5(c). In this plot the R_0 value is fixed at 1.0 to isolate the effects of varying the PDP reliability. The data series are produced by calculating the difference in hit rate percentage between precise and SAAM recycling. The graph shows that SAAM recycling can improve a systems reliability even in cases where the R_{PDP} is already high. Although the magnitude of improvement is low in the case where R_{PDP} is set to 0.99 it can be argued that highly reliable systems exist in environments where even small improvements in reliability are desirable.

We wish to understand the effect of the size of the BLP lattice on the hit rate of the SAAM component. We hypothesize that increasing the number of subjects and objects that share a particular security level in the lattice will increase the hit rate observed. We refer to this relationship as density.

There are two ways to test this hypothesis. The first involves fixing the subject and object population and varying the size of the BLP lattice. This is shown in Figure 5(b). It can be seen that the largest improvement in hit rate is experienced when the size of the lattice is small relative to the subject and object population. Another way to say this is that the hit rate improves as the density increases.

The second way to test this hypothesis is to fix the size of the lattice and vary the subject and object population. This is an area of ongoing research. Initial results indicate that higher hit rates are experienced; however, more investigation is required.

We also wish to understand the effect of the shape of the BLP lattice on the hit rate of the SAAM component. We propose to fix the density of subjects and objects and vary the shape of the lattice. The details of the experiments required to determine this relationship are still being refined.

B. *Latency Measurements*

Two latency graphs are presented. The first plot, Figure 6(a), shows the computation time for each of the components of the system. This is meant to demonstrate the relative complexities of the internal algorithms; however, the small size of the subject and object population limit the relevance of this graph. Studies of the computation time using larger subject and object populations are ongoing.

Figure 6(b) was generated using the latency calculations shown in Equation 4 for the same data set as previous plots. In this graph the round trip time between the PEP and PDP is varied to show the

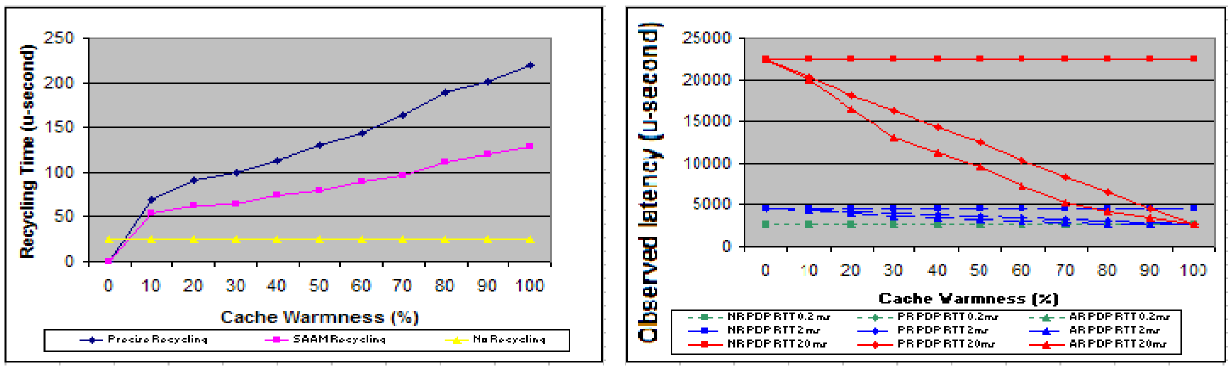

Fig. 6. (a) shows the the computation time for the recycling modes varies with cache warmness. (b) shows the average observed latency at the client site for three different network environments. The PDP is simulated to be farther away from the PEP by increasing the RTT between PEP and PDP.

effectiveness of recycling under various network deployments. Using a RTT of 20ms between PDP and PEP both recycling techniques achieve an order of magnitude improvement in observed latency at the client.

IV. Conclusion

Although this study is a work in progress, several points have already become clear. Applying SAAM recycling to a system using the BLP access control model can yield an improvement in reliability and latency under certain circumstances when compared to precise recycling. Both the latency and reliability improvements offered by SAAM recycling are dependant on the hit rate of the SAAM component. Effort should be focused on maximizing this value. Further, the system is only applicable in environments where a reasonable hit rate can be achieved.

The mapping of subjects and objects to security levels must be dense. If this is not the case the hit rate of the SAAM component degrades to match the hit rate of a precise recycling component.

The cache warmness must be kept high. If cache entries are constantly being invalidated then the hit rate will suffer accordingly.

Further work is necessary to understand the impact of lattice shape and population size on the hit rate. It is expected that in an environment with hundreds of thousands of subjects and objects the hit rate improvement offered by a SAAM recycling component will be more pronounced than current results have indicated.

References

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

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

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

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

[5] K. Beznosov. Flooding and Recycling Authorizations. In *New Security Paradigms Workshop (NSPW) September 2005.* Elec. and Comp. Engineering, University of British-Columbia, March 2005.

[6] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, Emmitt Witchel. Seperating Key Management from File System Security. In *17th ACM Symposium on Operating Systems Principles 1999*

[7] K. Beznosov. Recycling Authorizations: Toward Secondary and Approximate Authorizations Model (SAAM), LERSSE-TR-2005-01, LERSSE, Dept. of

[8] D. Bell and L. LaPadula. Secure Computer Systems: Mathematical Foundations,” Technical Report MTR-2547 Vol. 1, MITRE Corporation, Bedford, MA (Mar. 1973).

[9] D. Bell and L. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation,” Technical Report MTR-2997 Rev. 1, MITRE Corporation, Bedford, MA (Mar. 1975).

[10] W. Leung, J. Krampton, K. Beznosov. Authorization Recycling in BLP System. LERSSE-TR-2005-11, LERSSE, Dept. of Elec. and Comp. Engineering, University of BritishColumbia, November 2005.

[11] J. Musa, A. Iannino, K. Okumuto. *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, 1987.