An Autonomic Framework for Integrating Security and Quality of Service Support in Databases

Firas Alomari
The Volgenau School of Engineering
George Mason University

Daniel A. Menasce
Department of Computer Science
George Mason University

Summarized by
Harsha Balachandran
Introduction

● Autonomic computing systems are self-* systems that improve certain QoS goals or maintain them.
● These large, complex and emerging e-business systems also require
  ○ heavy database interaction
  ○ an increase in number of security mechanisms.
● Previous work focused on both individually.

**Problem**: Cannot compromise either as it requires a system administrator to specify the required QoS and security parameters separately and monitor them.

**Solution**: Autonomic controller

● using Intrusion Detection and Prevention Systems (IDPSs).
Intrusion Detection and Prevention Systems (IDPSs)

➔ Determine the normal behavior of users accessing the database. Any deviation is treated as intrusion.

➔ There are two main models of IDPSs:
  ◆ **anomaly detection**: Compares user’s normal behavior and their session data.
    ● A profile is built using queries issued to the database or the results of the queries executed.
  ◆ **misuse detection**: Compares user’s session with signatures of known attacks previously used.

➔ Challenges:
  ◆ System accuracy represented by false positives and false negatives.
  ◆ Solutions usually target specific scenarios and treat users uniformly.
  ◆ Impact on QoS requirements. Combining techniques compromises QoS.

Solution: Use Autonomic controller to integrate QoS and security by dynamically varying security configurations based on workload.
Framework : Environment

- Combination of IDPSs to achieve max security.
- **Syntax-centric techniques**
  - use query itself to evaluate the transactions submitted to the database.
  - evaluates requests before they are submitted to the database.
- **Data-centric techniques**
  - use query results to evaluate the transaction legitimacy.
  - evaluates query results before they are passed back to the users.
- Overall response time of the system depends on the workload intensity, the number and type of security mechanisms used, and the overhead associated with each mechanism.
Framework: Environment

➔ To deal with large number of users.
   ◆ Classify users into roles and specify either a profile/behavior for each role.
   ◆ Identify attack categories.
      ● specific scenario targeted.
      ● type of users targeted.
   ◆ Evaluate effectiveness of defense mechanism for each said category and overhead.
   ◆ Catalog of attack probabilities can be built by system analysis of database threats.
➔ Goal: Find max security level without compromising on QoS.
Framework: Environment

- Incoming requests are labeled according to the user’s role.
- The autonomic controller evaluates requests and send them to one or more of IDPSs.
- The controller is executed at certain intervals to
  ◆ obtain performance values.
  ◆ identify possible configurations that can be applied.
  ◆ compute performance and security metrics
  ◆ recommend a combination for the highest security level that meets the QoS requirements for a given workload.
- System is reconfigured to apply the new security policy to incoming requests.
Notation and Controller Policy Definitions

- $N = K + M$ is the total number of Security mechanisms.
  - $K$ is the number of syntax centric mechanisms.
  - $M$ is the number of data centric mechanisms.
- $R$ is the number of user roles.
- $A$ is the number of attack categories.
- $a_{r,j}$ is the likelihood of observing attack $j$ by role $r$.
- $d_{i,j}$ is the detection rate of attack $j$ by security mechanism $i$.
- $o_i$ is the overhead for mechanism $i$.
- The policy for role $r$ is $\rho_r = (\varepsilon_{r,1}, \ldots, \varepsilon_{r,i}, \ldots, \varepsilon_{r,N})$, where $\varepsilon_{r,i} = 0$ if security mechanism $i$ is not used for role $r$ transactions and equal to 1 otherwise.
- The overall system policy is then characterized by the vector $\vec{\rho} = (\rho_1, \ldots, \rho_R)$.
Utility Function

- Security utility for each role is a function of security mechanisms applied to the role. \( U_r^S(\rho_r) = f(\rho_r) \)
- Detection probabilities of each mechanism is different. The resulting utility is simplified using exponential averaging to meet the design. \( U_r^S(\rho_r) = \sum_{j=1}^{A} a_{r,j} \left( \ln \sum_{i=1}^{N} e^{d_{i,j} \times \varepsilon_{r,i}} \right) \).
- Total security utility is the weighted sum of all roles where \( w_r^g \) models risk and preferences for a role.
  \[
  U_{total}^S(\bar{\rho}) = \sum_{r} w_r^g \sum_{r} U_r^S(\rho_r),
  \]
- Total response time utility is the weighted sum of all response time utility functions.
  \[
  U_{total}^T(\bar{T}) = \sum_{r} w_r^t \sum_{r} U_r^T(T_r).
  \]
- Global utility can be written as a function of the policy, mechanisms overhead, workload and attack likelihood matrix.

The global utility function is a function of all utility functions:

\[
U_g(\bar{\rho}, T) = \alpha \times U_{total}^T(\bar{T}) + \beta \times U_{total}^S(\bar{\rho}),
\]

where \( \alpha \) and \( \beta \) are weights associated with response time and security, respectively, and \( \alpha + \beta = 1 \) and \( \alpha, \beta \geq 0 \).
Controller Architecture

- Uses heuristic search technique to find a near optimal configuration that maximizes the global utility function and meet the QoS and security requirements.
- The search for a new configuration is based on a greedy hill climbing-method.
- Sometimes high workloads, due to high demands or malicious attacks, result in a policy with no security mechanisms.
  - add constraints to performance to guarantee minimum security level irrespectively.
Queuing Network Model

- Computes QoS values/performance for a given configuration.
- IDPSs mechanisms are modeled concurrently.
- Even though mechanisms can be executed concurrently, the controller must wait for all of them to finish before deciding on the policy.
- The values of approximation for average response time for validate the controller’s ability and usefulness.
Experimental Evaluation & Results

Workload Intensity

System Utility

Average Arrival Rate Per Second

Utility Value

Time (minutes)
Experimental Evaluation & Results

**Difference in Global Utility**

- Controller $U_g$
- Static Security Policy $U'_{g}$
- No Security $U'_{g}$

**Model Accuracy**

- Predicted $U_{g}$
- Measured $U_{g}$

Graphs show the system global utility over time, comparing different scenarios and their impact on utility changes.
Experimental Evaluation & Results

Controller Response Time Per Role

Response Time

Average Response Time (Seconds)

- Static Security
- Controller
- No Security

Role 1
Role 2
Related Work & Conclusion

- Other works include almost everything used in this paper but independently.
- Geared towards specific products and scenarios
- Rely on system administrator to integrate and configure the system.

Review

- Paper addresses QoS and security requirements jointly even though they may be conflicting.
- Policy improves to maximize utility.
- Controller estimates performance impact of security policies and makes assignment decisions dynamically.
Future Work

- Exploring more utility expressions that better capture the system’s security requirements.
- Accuracy of the IDPSs, does not consider false positive and negatives and their significance.
- Evaluating using real data.
- Controller can be extended
  - to predict workloads.
  - include dynamic intervals for varied workload.
  - evaluate other heuristic techniques.