Use of AOP in Web Application Security

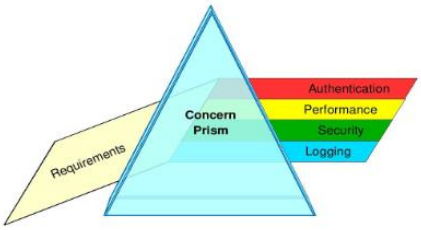
AProSec Aspect

As the Internet users increase, the need to protect web servers from malicious users has become a priority in many organizations and companies. With Aspect Oriented Programming (AOP), separating concerns when designing an application fosters reuse, parameterization and maintenance.

AProSec is a security aspect that deals with SQL Injection and XSS Cross Site Scripting.

AOP has been proposed as a technique for improving concerns separation in software systems and for adding crosscutting functionalities without changing the business logic of the software. AOP provides specific language mechanisms that make it possible to address concerns, such as security, in a modular way.

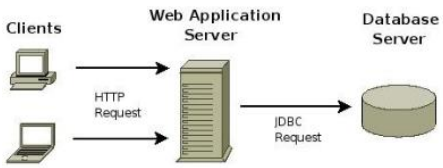
The purpose of AOP is to provide a solution for **code scattering** and **code tangling.** An analogy of how the different concerns can be separated from the requirements using AOP, is how a prism separates a light beam into a spectrum of colors.



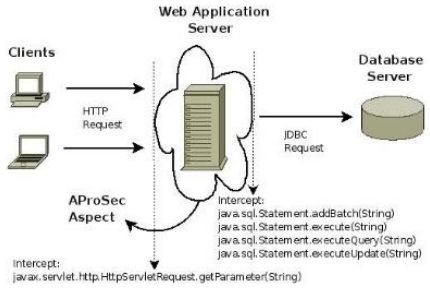
Authentication, performance, security and logging are commonly used concerns and are always spread through the application's code.

AOP, as a new programming paradigm, introduces notions such as an aspect, a join point, a point cut and an advice code. However, these notions do not replace existing ones such as a class, an object, a procedure or a method. Aspect oriented extensions exist for many languages, object oriented or procedural. Aspects can be applied at compile time or at runtime.

The picture below shows how a web application server (WAS)interacts with clients and database servers. When the client sends a request, it goes directly to the WAS and if it isn't validated, it can cause some unexpected behavior. When needed, the WAS forms a new request using the unvalidated parameters from the client, forwarding the attack to the database server. This is how most of the attacks are done, using unvalidated entries.



The next picture shows how AProSec protects the application by intercepting and validating all the requests from the client to the WAS and from the WAS to the database server. This prevents that any request goes unvalidated and that the attacks can get through. As shown in Fig. 3, AProSec surrounds the application, without having to change the application's source code.



The **AProSec** aspect can be used by any AOP framework and is composed of three parts. First, an advice (the added code) defines the validation process. Second, the way AProSec validates the requests depends on the options that the administrator selects on the configuration file. Finally, the point cut part (where the code is added) allows the weaving with the web application.

The advice part consists in two main validations: 1. HTTP request parameters and DB queries. In the HTTP requests, we validate the parameter value to avoid code injection and invalid HTML tags. For DB queries, the validation is made by analyzing the query string to prevent “always true” comparisons, semicolons and comments.

Even though single and double quotes are part of the SQL injection, the AProSec aspect manages them separately. There are defined all the validations that can be done, but the administrators can decide which ones to use by using the configuration file.

Regarding the weaving, AspectJ is the most widely used language for aspect oriented programming. It defines an extension of the Java programming language for dealing with aspects. The AspectJ compiler handles Java source code or byte code, weaves them with aspects, and generates some byte code that can then be executed with a standard Java virtual machine.

The best way to be protected against SQL attacks is to inspect all the data the user introduces to the application. Most of the work in this area attempts to limit the way in which a preprogrammed query will be used, allowing only the sentence that the programmer wants to define.

REFERENCES

- Gabriel Hermosillo, Roberto Gomez, Lionel Seinturier, Laurence Duchien. Using Aspect Programming to Secure Web Applications. Journal of Software, Science in China Press, 2007, 6 (2), pp.53-63. inria-00202894