Chapter 8.9
Aspect-Oriented Programming and Aspect.NET as Security and Privacy Tool for Web and 3D Web Programming

Vladimir O. Safonov
St. Petersburg University, Russia

ABSTRACT

This chapter covers the use of aspect-oriented programming (AOP) and Aspect.NET, an AOP toolkit for the.NET platform, to implement Web and 3D Web security and privacy. In this chapter the author shows that AOP is quite suitable as a trustworthy software development tool. AOP and Aspect.NET basics are overviewed using simple examples. Principles of applying Aspect.NET for Web and 3D Web security and privacy implementation are also discussed. The chapter presents a library of sample aspects implementing security and privacy for Web programming.

ASPECT-ORIENTED PROGRAMMING AND ITS USE FOR TRUSTWORTHY COMPUTING

Aspect-oriented programming (AOP) (Safonov, 2008) is a prospective approach to software development and maintenance. It is based on the concept of a *cross-cutting concern*. A *concern* in software design is an idea, consideration, or task. A *cross-cutting concern* is a concern that cannot be implemented as a hierarchy of modules – classes or procedures, and, therefore, whose implementation requires injection and activation of lots of new fragments of code (statements or definitions) scattered throughout the code of the target application.

DOI: 10.4018/978-1-61350-323-2.ch8.9

Copyright © 2012, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Typical examples of cross-cutting concerns are logging, security and privacy (in particular, in Web and 3D Web applications), since the implementation of all of them requires injecting into the target application code of many specific API calls to implement logging (e.g., tracking method calls and returns), security (e.g., permission checks), and privacy (e.g., encoding / decoding of information).

Currently most of the AOP tools are implemented for the Java platform. The most popular of them is AspectJ (AspectJ, 2001) - an extension of the Java language by AOP constructs and features. As for.NET, most of the existing tools for that platform are at research and experimental stage.

Aspect in AOP is an implementation of a cross-cutting concern. Aspect is considered as a new kind of *module* with specific ways of definition and use. In our terminology (Safonov, 2008), an aspect consists of a set of *actions*, specified in the aspect definition, each of them accompanied by its *weaving condition*. The purpose of the aspect is to reuse its actions in a variety of target applications by *weaving*. *Weaving* the aspect is the process of combining the code of the target application with the code of aspect in such a way that the actions of the aspect be activated in the *join points* of the target program selected by the *weaver* (part of the AOP toolkit) using the aspect's *weaving rules*. For example, a weaving rule in a security aspect can prescribe to weave a call of the security action *PermissionCheck()* before each call of the method *ResourceUpdate()*, wherever the latter call is found in the target application. Weaving is performed automatically by the specific component of the AOP toolkit, referred to as *weaver*.

Another example is related to privacy. Suppose that the method *TransferPrivateData()* of the target application is responsible for the transfer of some private data over the network, and the method *UsePrivateData()* implements the use of the private data by a network client. To protect the data from the attack, it is reasonable to encrypt them before the transfer, and decrypt after the transfer. If this encryption / decryption functionality is not implemented in the original version of target application (which is often the case in software practice), there is no need to change the target application's code manually, which is error-prone. It is possible to use AOP and to develop an aspect with two actions – *Encrypt()* and *Decrypt()*, such that the weaving rules of the aspect prescribe to insert the call of *Encrypt()* before each call of *TransferPrivateData()*, and the call of *Decrypt()* – after each call of the *UsePrivateData()*.

Even such simple example leads us to the conclusion that AOP could be a very powerful tool to implement security and privacy. Much more examples of using AOP for the purpose of trustworthy computing are provided in (Safonov, 2008). This chapter discusses how to implement these ideas in practice for Web and 3D Web programming, using our AOP toolkit for the.NET platform – Aspect.NET.

In general, as stated and explained in (Safonov, 2008), AOP methods appear to be generically related to application of *trustworthy computing* (TWC, 2002) – the approach initiated by Microsoft in 2002. Trustworthy computing (TWC) is a set of principles how to make the application code satisfying the *pillars* of TWC – *security, privacy, and reliability*. In software practice, in initial versions of many software products, such issues are often ignored or not paid enough attention to, for a number of reasons, e.g., lack of time, lack of secure code development experience, and so on. As the result, the software product appears to be vulnerable to attacks and unreliable. So, the product development team has to solve TWC problems of the product later, during its maintenance, as part of fixing bugs or implementing requests for enhancement. It takes a lot of time and human resources to examine the existing product code and try to add to it manually some new non-trivial security, privacy, or reliability functionality. So, in practice, the problem of *improvement* (or *modernization*) of the existing product code to make it more secure and reliable appears to be very important. AOP is

41 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/aspect-oriented-programming-aspect-net/61038

Related Content

Tampering Localization in Double Compressed Images by Investigating Noise Quantization
Archana Vasant Mire, Sanjay B. Dhok, Naresh J. Mistryand Prakash D. Porey (2020). *Digital Forensics and Forensic Investigations: Breakthroughs in Research and Practice (pp. 336-353).*
www.irma-international.org/chapter/tampering-localization-in-double-compressed-images-by-investigating-noise-quantization/252698

Data Privacy and Legal Considerations in Cyber Forensics
(2025). *Exploring the Cybersecurity Landscape Through Cyber Forensics (pp. 347-376).*
www.irma-international.org/chapter/data-privacy-and-legal-considerations-in-cyber-forensics/370619

An Incremental Acquisition Method for Web Forensics
Guangxuan Chen, Guangxiao Chen, Lei Zhangand Qiang Liu (2021). *International Journal of Digital Crime and Forensics (pp. 1-13).*
www.irma-international.org/article/an-incremental-acquisition-method-for-web-forensics/284502

Two Variations of Peer Intermediaries for Key Establishment in Sensor Networks
Jingyuan Rao, Min Tuand Xuanjin Yang (2020). *International Journal of Digital Crime and Forensics (pp. 1-14).*
www.irma-international.org/article/two-variations-of-peer-intermediaries-for-key-establishment-in-sensor-networks/252864

Exploring Artificial Intelligence (AI) in Forensic Pathology and Autopsy Analysis
Rishabha Malviya, Ashima Jain, Sahil Lal, Manmeet Kaur Aroraand Santosh Kumar (2025). *Forensic Intelligence and Deep Learning Solutions in Crime Investigation (pp. 125-146).*
www.irma-international.org/chapter/exploring-artificial-intelligence-ai-in-forensic-pathology-and-autopsy-analysis/371339