A Taxonomy Of Aspect-Oriented Security

Keshnee Padayachee, (Email: padayk@unisa.ac.za), University of South Africa, South Africa
Ntobeko Wakaba, (Email: notobeko@vioto.com), University of South Africa, South Africa

ABSTRACT

Aspect-Oriented Programming is gaining prominence, particularly in the area of security. There are however no taxonomies available, that classify the proliferation of research done in the area of Aspect-Oriented Security. This paper attempts to categorize research outputs conducted in this area, and evaluate the usability of the aspect-oriented paradigm in terms of software security.

Key Words: Taxonomy, Aspect-Oriented Programming, Software Security

1. Introduction

Although aspect-oriented programming is not yet ubiquitous in industry, it is receiving considerable attention from research and practitioner communities such as IBM, Northeastern University in the United States, University of Twente in the Netherlands and Xerox (Miller, 2001). The field has matured to such an extent that it generated its own conference. The first international conference on aspect-oriented programming was held in Twente, Netherlands in 2002 (Padayachee and Eloff, 2006). Aspect-oriented programming is gaining prominence in all areas of software development, particularly in the area of software security. Several authors have been citing the benefits of using aspect-oriented programming to implement security concerns (see (De Win et al., 2002c) and (Viega and Evans, 2000))**.** Aspect-oriented software development 'is relevant for all major pillars of security: authentication, access control, integrity, non-repudiation as well as for supporting the administration and monitoring disciplines required for effective security' (Bodkin, 2004). There are however, no taxonomies available that classify the proliferation of research done in the area of aspect-oriented security. This paper attempts to categorise research outputs conducted in this area, and evaluate the usability of the aspect-oriented paradigm in terms of software security.

The principal argument supporting the utilization of aspect-oriented programming to develop secure software is that security concerns tend to crosscut objects, resulting in the code tangling phenomenon. For example, software tampering (Falcarin et al., 2004) and encryption (Boström, 2004) may be considered as crosscutting concerns, given that these concerns tend to crosscut across several components in a system. Aspect-orientation has the potential to enhance the implementation of security concerns, in terms of reusability and extensibility, thereby improving the robustness and maintainability of a system. Evidently, abstracting a security feature into a security aspect increases the possibility that it may be reused for other applications. Access control and encryption, for example, have similar requirements for most applications. Vanhaute and De Win (2001) have demonstrated how to convert these security concerns into reusable generic aspects. It has also been shown that aspect-oriented software design is flexible enough to accommodate the implementation of additional security features after the functional system is developed. Laney et. al (2004) have demonstrated that the aspect-oriented paradigm facilitates the implementation of additional security features to a legacy system, without modifying the existing code. The level of abstraction offered by aspect-orientation may also facilitate the separation of roles between application developers and security specialists (Bodkin, 2004), which can improve the efficacy of the software development process.

This paper attempts to categorise research outputs conducted using the aspect-oriented paradigm to facilitate the development of security concerns into a software system. This research serves to inform researchers and business communities alike, about the growing potential of the aspect-oriented paradigm, in terms of developing secure software. The subsequent two sections focus on aspect-oriented programming, and its influence on software security. Section four, elaborates on taxonomy developed. Section five concludes with some insights for future work and final comments.

2. Background On Aspect Programming

The object-orientated paradigm was found to be inadequate in terms of design and implementation of crosscutting concerns, as there is no elegant way of modularising these crosscutting concerns. Aspect-oriented programming provides explicit language support for modularising design decisions that cross-cut a functionally-decomposed program (Walker et al., 1999), allowing the developer to maintain the code (crosscutting functionality) in a modularised form. It is important to note that aspect-orientation maintains all the benefits of the object-oriented paradigm and should be viewed as extension rather than a replacement of object-oriented technologies.

An aspect-oriented system consists of two concepts: components and aspects. The components form the atomic and loosely coupled concerns of the system, and the aspects implement additional crosscutting functionality of the system. According to Kiczales et al. (2001), aspects consist of the following constructs:

- *joinpoints* or *pointcut designators*: these are definitions of interception points in the system where the aspect could possibly take action.
- *advice*: an advice defines the functionality to be executed before, during, or after a defined joinpoint, or even possibly instead of the defined joinpoint.

An aspect weaver is then used to combine both component and aspect programs into a final program.

According to Elrad et al. (2001), the advantages gained by using aspect-oriented programming are as follows:

- Increased modularity: the separation of cross-cutting concerns and their capture in a single class.
- Less code needs to be written: applications may be developed faster.
- Easier maintenance: code is no longer scattered throughout an application but localized.
- Easier code reuse: since cross-cutting concerns are localized, they may be easily reused.
- Easier understanding, while maintaining optimisation: the code is no longer tangled, thereby allowing easier comprehension.
- More flexibility: with aspects, there are more ways to code the same concern.
- Reduced inheritance anomalies: the use of aspects allows a simpler class hierarchy.

In addition to the benefits of reduced complexity and improved maintainability of software, programmers may be better able to understand an aspect-oriented program when the effect of aspect code has a well defined scope The presence of aspect code may also alter the strategies programmers use to address tasks perceived to be associated with aspect code (Walker et al., 1999).

Despite the benefits offered by aspect-orientation, there are some drawbacks. Alexander & Bieman (2002) maintain that as a result of the weaving process, the isolation of faults will be difficult, as faults may reside in the source code, the aspect, or the woven code. Another challenge, as noted by Chen (2004) is that of understandability – a many-to-many relationship may exist between aspects and the primary abstractions they integrate with, thus potentially requiring understanding of many other aspects to understand only one. A complication may arise when a collection of aspects to be woven are written by several different programmers. Each programmer must have the knowledge of the set of primary abstractions that their aspects can be woven with hence each programmer must know about the other aspects that they make use of, either by direct composition or indirectly as a result of weaving. All of these activities will pose more cognitive burden on the aspect developer (Chen, 2004).

3. Establishing Categories In Aspect-Oriented Security Research

There has been a proliferation of research articles within aspect-oriented security within recent years. The evidence of an intrinsic relationship between aspect-orientation and the development of secure software is so substantial that De Win et al. (2003) have attempted to generalize these specific security-related concerns into a framework. While Georg et al. (2002) have proposed a technique to model and integrate security concerns into designs using the aspect-oriented paradigm by regarding security aspects as design patterns. The subsequent discussion, explores the relevancy of aspect-oriented technology in terms of some of the fundamental security concerns such as access control and authentication, cryptographic controls, data protection and information flow controls.

3.1 Access Control And Authentication

Access control is a fundamental part of computer security where every requested access must be governed by an access policy stating who is allowed access to what; the request must then be mediated by an access policy enforcement agent (Pfleeger and Pfleeger, 2003). The seminal work in this area was conducted by De Win et. al (2002b) where they have actually generalized the aspects they developed for access control, to promote the reusability of these aspects. Slowikowski and Zieliński (2003) considered how aspect-oriented security could enhance container-managed security. They have also demonstrated how identification, authentication, and access control may be applied to components without modifying the components. They concluded that aspect-oriented security required no modification of the application's sources to introduce security and that the procedure is highly flexible and extensible. The significance here is that if access policies are unknown or vague, the access control features may be implemented after the development of other requirements, or when these policies have been clearly defined. Further, the abstraction of access control policies makes the management and development much easier, as security experts may be allocated specifically to the development of these features. Recently Ramachandran et al. (2006) also addressed authentication and authorization within the aspect-oriented paradigm, but they provided a more generic approach. Where explicit naming is forfeited in favor of more generic pointcut designators, the aspect-oriented paradigm allows a more generic implementation of the access control through the use of wildcards. Using wildcards eliminate the need for explicit naming (Kiczales et al., 2000).

The aspect-orientated paradigm's versatility in terms of access control measures has been further validated by studies conducted within differing approaches to access control. The paradigm has been leveraged to implement discretionary access control (see (De Win et al., 2002a)), role-based access control (see (Pavlich-Mariscal et al., 2005a)) and mandatory access control (see (Ramachandran et al., 2006)).

3.2 Cryptographic Controls

Cryptographic controls such as encryption help to protect the confidentiality, authenticity, and integrity of information (Fiedler, 2002). In an experiment conducted by Boström (2004), it was found that database encryption can be added after the initial system was already built using aspect-oriented programming; This case study showed that using aspect-oriented programming resulted in better modularity, database independence, and less code, but there are instances where the logic developers cannot be totally alienated from the process of encryption because sometimes developing the functionality is dependent on the encryption process.

3.3 Information Flow Controls

These are methods of regulating the dissemination of information among objects throughout the system (Denning and Denning, 1977). There have been several studies on access control, encryption, and security-related bugs within the aspect-oriented paradigm, but only one related work on information flow control within the aspect-oriented paradigm by Kawachi and Masuhara (2004). As this study is specifically on sanitizing, the authors do not address security classifications and their dataflow definition "only deals with direct information flow". This article shows potential for the aspect-oriented programming language to be extended for security purposes by identifying new constructs to specify pointcut designators. As the aspect-oriented language is still evolving, these types of security enhancements could be incorporated in future releases to further promote its relevance to security.

3.4 Protection From Invasive Software

The ACM classification system limits this category to viruses, worms, and Trojan horses. The taxonomy devised here extended this category to imply any malicious modification to software as well. Palmer (2002) considered dynamic aspect-oriented programming in an untrusted environment, where underlying components may be damaged by potentially hostile aspects. This research does not use the aspect-oriented paradigm to verify whether an aspect is hostile or not. Aspect-oriented programming has however, been used to implement software tampering detection mechanisms in applications running on untrusted hosts (Falcarin et al., 2004). Here, an aspect-oriented program was used to realize self-checking. Self-checking is a process where a program checks itself to verify that it has not been modified. In terms of evolving verification techniques, as security threats change, using the separation of concerns principles makes it easier to 'swap in and out and evaluate alternative treatment options' (Houmb et al., 2004).

3.5 Security Kernels

A security kernel is responsible for enforcing the security mechanisms of the entire operating system (Pfleeger and Pfleeger, 2003). Engel and Freisleben (2005) developed a tool for deploying dynamic aspects in the kernel space of an operating system which included enhancing security features within the operating system level. Due to rapid adaptation of security kernels caused by dynamically changing requirements, internal conditions, and crosscutting functionality, Engel and Freisleben (2005) established that dynamic aspect-oriented programming is highly suitable for this type of enforcement. Additionally, the performance impact of using aspect-orientation was negligible in most instances.

3.6. Verification

Typically, a security specialist wants to be certain that a given program computes a particular result, computes it correctly, and does nothing more. Validation is a more general term; It includes verification and less rigorous methods (Pfleeger and Pfleeger, 2003). Grundy and Ding (2002) have developed "validation agents" that use aspect characterizations to verify that software components meet constraints in actual deployment situations. These test both functional and non-functional components, such authentication and encryption. Kumar et al. (2001) developed a framework that uses the aspect-oriented programming paradigm to verify that the commercial-off-the-shelf components are developed as per security contract. Validating a component's performance, resource usage, □transaction support, security, data persistency, and distribution, are issues that crosscut a system's components hence the aspect-oriented paradigm is appropriate for these purposes. Furthermore, as verification procedures tend to be similar in most applications, using the aspect-oriented paradigm facilitates the reusability of these validation measures (Grundy and Ding, 2002) via abstraction.

3.7 Security Software Engineering

This category represents the papers that had a strong theme of security software engineering practices and techniques. Security concerns must inform every phase of software development, from the requirements to the design, implementation, and deployment (Devanbu and Stubblebine, 2000). The contributions here are related to the engineering of secure and trustworthy software systems. There is often a confusion between security software and software security; The former entails a set of add-on features such as cryptography, while latter is an "emergent property" of a complete system (Mcgraw, 2002) Software security involves designing, analyzing, and coding for security, over the whole life cycle. The flexibility of the aspect-oriented paradigm allows these concerns to be dealt with in a more systematic way via abstraction. Hence the programming complexity is largely reduced.

During systems development, it is probable that security requirements may not be clearly defined, or may change during system development, and quite possibly after the system is built. The extensibility of the aspect-oriented paradigm allows these concerns to be implemented after other requirements have been developed. Further, if a security concern had to be maintained due to the discovery of new security threats in the environment, separations of concerns would fundamentally simplify this task. Within other programming paradigms, it would have possibility resulted in several fixes, across several components, thus resulting inconsistencies and regression faults.

In terms of developing code that is secure, both Shah and Hill (2003b), and Viega and Evans (2000) explored security-related bugs. Shah and Hill(2003a) developed *An Aspect-Oriented Security Framework* which focused on avoiding security-related bugs such as buffer overruns, time-of-check-to-time-of-use, format string vulnerabilities, protection of communication channels, event ordering enforcement, and type safety. These types of security concerns are highly pervasive, and the typical penetrate-and-patch approach is highly ineffective. Although Viega and Evans(2000) have not specifically used aspect-oriented tools, their methodology closely parallels the aspect-oriented philosophy.

4. Derivation Of The Taxonomy

In developing the taxonomy for aspect-oriented security, several existing computing and security related taxonomies were reviewed. Compared to the other taxonomies reviewed, the ACM Computing Classification System has the simplest structure. Section D.4.6 of the ACM Computing Classification System relates to software security and protection in particular. Although taxonomies such as the 'Taxonomy for Information Security' by Meadows (1993) offer much richer classification content than section D.4.6 of the ACM Computing Classification System, it is not appropriate at this stage, as the literature in the area of aspect-oriented security has not matured to such an extent. Other taxonomies reviewed were not specifically related to the area under consideration. For instance, the "Taxonomy of computer program security flaws" by Landwehr et al. (1994) was not suitable as it considers security flaws in operating systems, and they have not categorised the flaws that occur in application programs. As aspect-oriented programming is used to develop secure applications, issues such as access control and other application dependant aspects, are taken into consideration. The ACM Computing Classification System was selected as the most appropriate taxonomy for this purpose. The ACM Computing Classification System (section D.4.6) offers several benefits over the other taxonomies as it has been proven and is in wide use, and allows for future extension of the classification system for richer classification content.

The presentation and techniques used to analyse the taxonomy is based on the work conducted by Hankerson (2003). Hankerson (2003) developed a taxonomy for Aspect-Oriented Programming in general. Initially, themes identified in the aspect-oriented security research (in Section 3) mapped very well to the security section of the ACM Classification System. It however became apparent that an additional section was required to categorise general security related software engineering practices, which encompassed specifying, designing, and implementing secure software with aspect-orientation. This category includes inadvertent security flaws, which impacts the implementation phase and deployment phase. These types of security flaws can be avoided if the principles of security engineering are applied. Additionally, aspect-oriented programming is often seen as means to avoid these kinds of security flaws. These flaws include buffer overruns, Time-of-check-to-time-of-use (TOCTTOU), and Format string vulnerabilities.

Since the number of papers to be collected was potentially large, a database was used to track and categorise all the articles. The database schema used for building the taxonomy is shown in Figure 1 above. As Figure 1 illustrates, there are two tables in the database schema: PUBLICATION and CATEGORY. The PUBLICATION table contains all the publication details of the papers collected. The primary key publication_id is an automatically generated sequence number. The CATEGORY table contains all the possible categories. The taxonomy on aspect-oriented security does not span multiple dimensions, as the number of articles collected made it feasible for only one level to exist; This level is called the Primary Dimension (Hankerson, 2003).

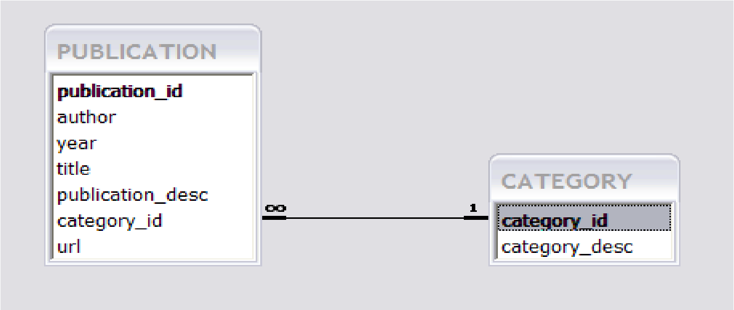

Figure 1: The taxonomy database schema

Most of the articles found had very definite themes and it was possible to categorize them into one of the categories listed. Four of the articles collected however, did have at least two very strong themes, which made them each fall into more than one category; this is explained by the total count of the articles in the categories (56) compared to the total count of the articles listed in appendix A (52).The 52 articles (listed in Appendix A) were categorised into the following eight categories:

(1) Access Controls
(2) Authentication
(3) Cryptographic Controls
(4) Information flow control
(5) Invasive software
(6) Security kernels
(7) Verification
(8) Security software engineering

The list represents the primary list of categories, as mentioned earlier. As the research in the field of Aspect-Oriented Security grows, the above list can be refined into further sub-categories (or dimensions) for deeper analysis.

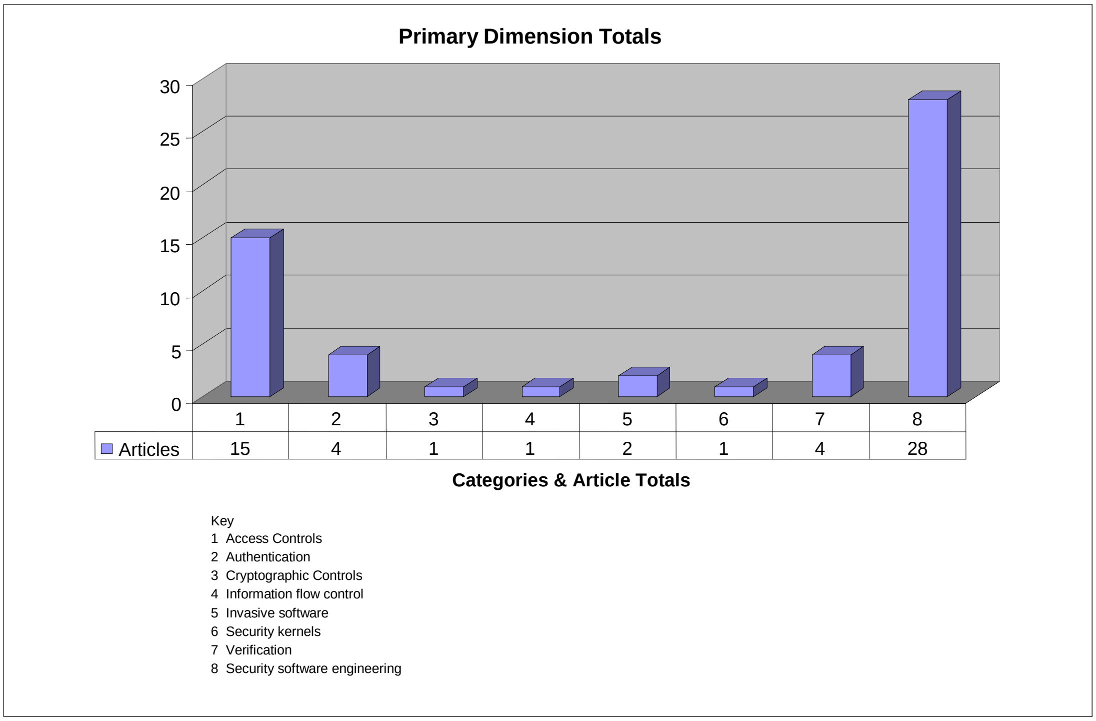

Figure 2: Primary Dimension totals represented in a graph

As graph (Figure 2) above indicates, the vast number of articles found had a strong theme of software engineering. As previously stated, it is also important to note that the category of software engineering is absent from the security section of the ACM Computing Classification System. Access Controls and Authentication have the second highest number of articles categorised. Verification and Invasive software follow with 3 and 2 articles respectively. Finally the Security Kernels, Cryptographic Controls, and Information Flow Control categories had one article each.

The following graph (Figure 3) represents the year breakdown of the articles collected. This helps to highlight the activity in the area of research over the years. Figure 3 illustrates graphically the number of articles collected that have been released between 2000 and 2005. A definite trend exists, in that the numbers of articles have been increasing steadily over the years. This is a strong indicator that aspect-orientation is highly relevant to the development of secure systems.

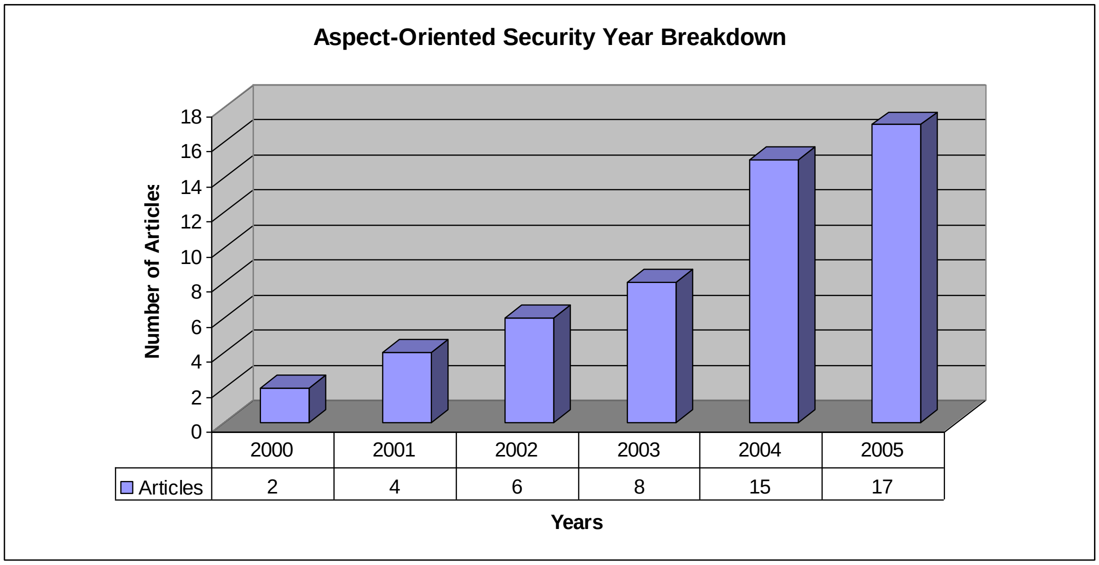

Figure 3: The number of Aspect-Oriented Security articles per year.

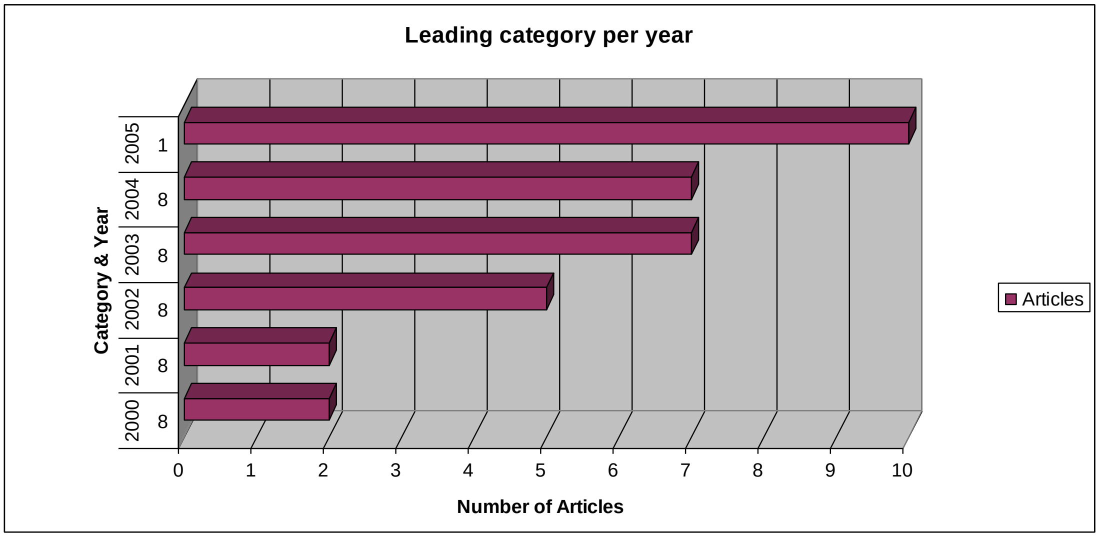

Figure 4: Graphical illustration of leading category per year

The final graph (Figure 4) concludes the graphical presentation of articles collected with a graph illustrating the leading category of articles per year. Figure 4 shows the leading category of articles per year of Aspect-Oriented Security articles. As depicted in Figure 4, category 8 (Security Software Engineering) dominates with the most articles collected in 2000 to 2004, while the year 2005 is dominated by category 1 (Access Controls). The graph also indicates that over the years, the most articles released were of categories 8 (Security Software Engineering) and 1 (Access Control) – in that order.

5. Conclusion

This research only provided a primary dimension on aspect-oriented security. Future work would naturally evolve into providing a secondary dimension on this area. This research is governed by time factors. It is obvious that since the completion of this work in 2005, more articles have emerged, but it was decided to confine the study to a specific epoch, to avert formulating premature pronouncements. Further, the list presented here is by no means exhaustive. From the research conducted, it is evident that there is a growing trend towards the use of the aspect-oriented paradigm with the security context. It was found that, in terms of aspect-oriented security, there was a proliferation of research in security engineering and access controls, while relatively little research was conducted in cryptographic controls, security kernels, and information flow controls. It is evident that aspect-oriented programming is expected to influence the development of secure software in the future. As such, there is a need for more evaluative studies, with respect to the usability issues of aspect-orientation, in terms of secure software development. Additionally, there is a need for research to be conducted, in terms of considering how aspect-oriented security teams work. It is assumed that security teams should be able to work on security issues without consulting the engineers that built the functional system. The notion of aspect-oriented security teams would be observed and reported on as well. Research on discovering new constructs to specify pointcut designators to further facilitate security, merits exploration. Aspect-orientation has been found to be highly complementary in the creation secure software, as security issues tend to crosscut components, coupled with the tendency of security concerns to evolve as new security threats emerge. Hence the flexibility and extensibility of the aspect-orientated paradigm is highly conducive to these types of modifications and extensions.

References

1. ALEXANDER, R. T. AND BIEMAN, J. M. 2002. Challenges with Aspect-oriented Technology. In *ICSE Workshop on Software Quality*, Orlando, Florida, 25 May 2002.
2. BODKIN, R. 2004. Enterprise Security Aspects. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004, 1-12.
3. BOSTRÖM, G. 2004. A case study on estimating the software engineering properties of implementing Database Encryption as an aspect. In Proceedings of the 3rd international conference on Aspect-oriented software development, Lancaster, UK, March 2004, 1-6.
4. CHEN, L.2004.Aspect-Oriented Programming in Software Engineering. Wake Forest University, Department of Computer Science.
5. DE WIN, B., JOOSEN, W. AND PIESSENS, F. 2002a Developing Secure Applications through Aspect-Oriented Programming. In *Aspect-Oriented Software Development*, Aksit, M., Clarke, S., Elrad, T. AND Filman, R. E. (Eds.), Addison-Wesley.
6. DE WIN, B., JOOSEN, W. AND PIESSENS, F. 2003. AOSD Security: A Practical Assessment. In Workshop on Software engineering Properties of Languages for Aspect Technologies (SPLAT03), Boston, Massachusetts, 17-21 March 2003, 1-9.
7. DE WIN, B., PIESSENS, F. AND JOOSEN, W. 2002b. On the importance of the separation-of-concerns principle in secure software engineering. In Workshop on the Application of Engineering Principles to System Security Design, Boston, Massachusetts, 6-8 November 2002, 1-10.
8. DE WIN, B., VANHAUTE, B. AND DE DECKER, B. 2002c.How aspect-oriented programming can help to build secure software. *Informatica,* 26(2)**,** 141-149.
9. DENNING, D. E. AND DENNING, P. J. 1977.Certification of Programs for Secure Information Flow. *Communications of the ACM,* 20(7)**,** 504 -513.

10. DEVANBU, P. T. AND STUBBLEBINE, S. 2000. Software engineering for security: a roadmap. In Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 4-11 June 2000, 227-239.
11. ELRAD, T. M., ASKIT, G., KICZALES, K., LIEBERHERR, H. AND OSSHER. 2001. Discussing Aspects of AAOP. *Communications of the ACM,* 44(10)**,** 33-38.
12. ENGEL, M. AND FREISLEBEN, B. 2005. Supporting autonomic computing functionality via dynamic operating system kernel aspects. In Proceedings of the 4th international conference on Aspect-oriented software development, Chicago, Illinois, 22-26 March 2005, 51 – 62.
13. FALCARIN, P., BALDI, M. AND MAZZOCHI, D. 2004. Software Tampering Detection using AOP and mobile code. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004, 1-6.
14. Software solutions for successful Management, On the necessity of management of information security.2002.[Online] Available: <http://www.noweco.com/wp_iso17799e.htm>,
15. GEORG, G., RAY, I. AND FRANCE, R. 2002. Using Aspects to Design a Secure System. In *Eighth IEEE* International Conference on Engineering of Complex Computer Systems (ICECCS'02), Greenbelt, Maryland, USA, 2-4 December 2002, 117-126.
16. GRUNDY, J. AND DING, G. 2002. Automatic Validation of Deployed J2EE Components Using Aspects. In 17th IEEE International Conference on Automated Software Engineering (ASE'02), Edinburgh, UK, 47-57.
17. HANKERSON, M. B.2003.Towards a Taxonomy of Aspect Oriented Programming, Masters Thesis. East Tennessee State University. Johnson City, Tennessee.
18. HOUMB, S. H., GEORG, G., FRANCE, R. AND MATHESON, D. 2004. Using aspects to manage security risks in risk-driven development. In 3rd International Workshop on Critical Systems Development with UML, Lisbon, Portugal, 12 October 2004, 71-84.
19. KAWAUCHI, K. AND MASUHARA, H. 2004. Dataflow Pointcut for Integrity Concerns. In AOSD 2004 Workshop on AOSD Technology for Application-level Security (AOSDSEC), Lancaster, UK, 22-26 March 2004, 1-6.
20. KICZALES, G., HILSDALE, E., HUGUNIN, J., KERSTEN, M. AND PALM, J. 2001.Getting Started with AspectJ. *Communications of the ACM,* 44(10)**,** 59-65.
21. KICZALES, G., HUGUNIN, J., KERSTEN, M., LAMPING, J., LOPES, C. AND W.G., G. 2000. Semantics-Based Crosscutting in AspectJ. In Workshop on Multi-Dimensional Separation of Concerns inSoftware Engineering (ICSE 2000).
22. KUMAR, A., SINGH, A. K. AND BABU, R. S. 2001.A security assurance framework for component based software development. *Informatica,* 25(4)**,** 509 - 515.
23. LANDWEHR, C. E., BULL, A. R., MCDERMOTT, J. P. AND CHOI, W. S. 1994.A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys,* 26(3)**,** 211-254.
24. LANEY, C. R., VAN DER LINDEN, J. AND THOMAS, P. 2004. Evolution of Aspects for Legacy System Security. In AOSD workshop on dynamic aspects AOSDSEC'04, Lancaster, UK, 22-26 March 2004. 1-7.
25. MCGRAW, G. 2002.Managing Software Security Risks. *IEEE Computer,* 35(4)**,** 99-101.
26. MEADOWS, C. 1993. An outline of a taxonomy of computer security research and development. In Proceedings on the 1992-1993 workshop on New security paradigms, Little Compton, Rode Island, August 1993, 33-35.
27. MILLER, S. K. 2001.Aspect-Oriented Programming Takes Aim at Software Complexity. *Computer,* 34(4)**,** 18-21.
28. PADAYACHEE, K. AND ELOFF, J. H. P. 2006. The Next Challenge: Aspect-Oriented Programming. In Proceedings of the Sixth IASTED International Conference on Modeling, Simulation and Optimization, Gaborone, Botswana, 11-13 September 2006, (Ed.)Nyongesa, H., ACTA Press, 304-307.
29. PALMER, D. 2002. Dynamic Aspect-Oriented Programming in an Untrusted Environment. In *International Symposium on Distributed Objects and Applications (DOA), Irvine, California, 28-30* October 2002.

30. PAVLICH-MARISCAL, J., MICHEL, L. AND DEMURJIAN, S. 2005. A Formal Enforcement Framework for Role-Based Access Control using Aspect-Oriented Programming. In Proceedings of ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (Models/UML 2005), Montego Bay, Jamaica, 2-7 October 2005, 537-552.
31. PFLEEGER, C. P. AND PFLEEGER, S. L. 2003. *Security in Computing.* Prentice Hall, Upper Saddle River, New Jersey,
32. RAMACHANDRAN, R., PEARCE, D. J. AND WELCH, I. 2006. AspectJ for Multilevel Security. In The 5th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Bonn, Germany, 2006, 21 March 2006.
33. SHAH, V. AND HILL, F.2003a.An Aspect-Oriented Security Assurance Solution, Defense Advanced Research Projects Agency, [Online] Available: http://www.stormingmedia.us/50/5039/A503914.html.
34. SHAH, V. AND HILL, F. 2003b. An Aspect-Oriented Security Framework. In *DARPA Information Survivability Conference and Exposition*, Washington D.C, 22-24 April 2003, 143.
35. SLOWIKOWSKI, P. AND ZIELINSKI, K.2003. Comparison Study of Aspect-oriented and Container Managed Security. AGH University of Science and Technology.
36. VANHAUTE, B. AND DE WIN, B. 2001. AOP, Security and Genericity. In 1st Belgian AOSD Workshop, Vrije Universiteit Brussel, Brussels, Belgium, 8 November 2001.
37. VIEGA, J. AND EVANS, D. 2000. Separation of concerns for security. In ICSE 2000 Workshop on Multi-Dimensional Separation of Concerns in Software Engineering, Limerick, Ireland, June 2000, (Eds.).Tarr, P., Harrison, W., Ossher, H., Finkelstein, A., Nuseibeh, B. AND Perry, D., 126-129.
38. WALKER, R. J., BANIASSAD, E. L. A. AND MURPHY, G. C. 1999. An initial assessment of aspect-oriented programming. In Proceedings of the 21st international conference on Software engineering, Los Angeles, California, May 1999, 120-130.

Appendix A

Table 1: List of categories and their corresponding articles

Category Number	Category	Articles
1	Access Controls	(Chen, 2005), (De Win et al., 2001), **(Hasle, 2004)**, (Koch and Pauls, 2005), (Laney et al., 2004), (Li et al., 2005), (Naqvi and Riguidel, 2005), (Pavlich-Mariscal et al., 2005a), (Pavlich-Mariscal et al., 2005c), **(Ren et al., 2005),** (Rits et al., 2005), **(Slowikowski and Zielinski, 2003),** (Song et al., 2005),(Verhanneman et al., 2005), (Xu and Goel, 2005)
2	Authentication	(Baligand and Monfort, 2004), **(Hasle, 2004), (Kuntze et al., 2005)** , **(Slowikowski and Zielinski, 2003)**
3	Cryptographic Controls	(Boström, 2004)
4	Information Flow Controls	(Kawauchi and Masuhara, 2004)
5	Invasive Software	(Falcarin et al., 2004), (Palmer, 2002)
6	Security Kernels	(Engel and Freisleben, 2005)
7	Verification	(Eichberg et al., 2004), (Grundy and Ding, 2002), (Kumar et al., 2001), (Xu and Nygard, 2005)
8	Security Software Engineering	(Bodkin, 2004), (Charfi and Mezini, 2005), (Choi, 2000), (De Win, 2004) , (De Win et al., 2002a), (De Win et al., 2003), (De Win et al., 2002b), (De Win et al., 2002c), (Fiege et al., 2004), (Fox and Jurjens, 2005), (Gao et al., 2004), (Georg et al., 2002), (Hayley et al., 2004), (Houmb et al., 2004), (Huang et al., 2004), **(Kuntze et al., 2005)**, (Laney et al., 2003), (Pavlich-Mariscal et al., 2005b), **(Ren et al., 2005)**, (Rits, 2003), (Robinson et al., 2004), (Shah and Hill, 2003a), (Shah and Hill, 2003b), (Shreyas, 2003), (Vanhaute and De Win, 2001), (Viega et al., 2001), (Viega and Evans, 2000), (Welch and Stroud, 2003),

1. ALEXANDER, R. T. AND BIEMAN, J. M. 2002. Challenges with Aspect-oriented Technology. In *ICSE* Workshop on Software Quality, Orlando, Florida, 25 May 2002.
2. BALIGAND, F. AND MONFORT, V. 2004. A concrete solution for web services adaptability using policies and aspects. In Proceedings of the 2nd international conference on Service oriented computing, New York, 15-19 November 2004, 134 – 142.
3. BODKIN, R. 2004. Enterprise Security Aspects. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004, 1-12.
4. BOSTRÖM, G. 2004. A case study on estimating the software engineering properties of implementing Database Encryption as an aspect. In Proceedings of the 3rd international conference on Aspect-oriented software development, Lancaster, UK, March 2004, 1-6.
5. CHARFI, A. AND MEZINI, M. 2005. Using Aspects for Security Engineering of Web Service Compositions. In Proceedings of the 2005 IEEE International Conference on Web Services, Orlando, Florida, 12-15 July 2005, 59-66.
6. CHEN, K. 2005. Using Dynamic Aspects for Delegating Fine-Grained Access Rights. In 12th Asia-Pacific Software Engineering Conference (APSEC'05), Taiwan, 17 December 2005, 783-789.
7. CHEN, L.2004.Aspect-Oriented Programming in Software Engineering. Wake Forest University, Department of Computer Science.
8. CHOI, J. P. 2000. Aspect-Oriented Programming with Enterprise JavaBeans. In Fourth International Enterprise Distributed Object Computing Conference (EDOC'00), Makuhari, Japan, 25-28 September 2000, 252-261.
9. DE WIN, B.2004.Engineering Application-level Security through Aspect-Oriented Software Development, PHD Thesis. The Netherlands, Katholieke Universiteit.
10. DE WIN, B., JOOSEN, W. AND PIESSENS, F. 2002a Developing Secure Applications through Aspect-Oriented Programming. In *Aspect-Oriented Software Development*, Aksit, M., Clarke, S., Elrad, T. AND Filman, R. E. (Eds.), Addison-Wesley.
11. DE WIN, B., JOOSEN, W. AND PIESSENS, F. 2003. AOSD Security: A Practical Assessment. In Workshop on Software engineering Properties of Languages for Aspect Technologies (SPLAT03), Boston, Massachusetts, 17-21 March 2003, 1-9.
12. DE WIN, B., PIESSENS, F. AND JOOSEN, W. 2002b. On the importance of the separation-of-concerns principle in secure software engineering. In Workshop on the Application of Engineering Principles to System Security Design, Boston, Massachusetts, 6-8 November 2002, 1-10.
13. DE WIN, B., VANHAUTE, B. AND DE DECKER, B. 2002c.How aspect-oriented programming can help to build secure software. *Informatica,* 26(2)**,** 141-149.
14. DE WIN, B., VANHAUTE, B. AND DECKER, B. 2001. Security Through Aspect-Oriented Programming. In Advances in Network and Distributed Systems Security, IFIP TC11 WG11.4 First Working Conference on Network Security, Leuven, Belgium, November 2001, (Eds.).Decker, B. D., Piessens, F., Smits, J. AND Herreweghen, E. V., Kluwer Academic Publishers, Boston, 125-138.
15. DENNING, D. E. AND DENNING, P. J. 1977.Certification of Programs for Secure Information Flow. *Communications of the ACM,* 20(7)**,** 504 -513.
16. DEVANBU, P. T. AND STUBBLEBINE, S. 2000. Software engineering for security: a roadmap. In Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 4-11 June 2000, 227-239.
17. EICHBERG, M., MEZINI, M., SCHAFER, T., BERINGER, C. AND HAMEL, K. M. 2004. Enforcing System-Wide Properties. In Australian Software Engineering Conference (ASWEC'04), Melbourne, Australia, 13-16 April 2004, 158-167.
18. ELRAD, T. M., ASKIT, G., KICZALES, K., LIEBERHERR, H. AND OSSHER. 2001. Discussing Aspects of AAOP. *Communications of the ACM,* 44(10)**,** 33-38.
19. ENGEL, M. AND FREISLEBEN, B. 2005. Supporting autonomic computing functionality via dynamic operating system kernel aspects. In Proceedings of the 4th international conference on Aspect-oriented software development, Chicago, Illinois, 22-26 March 2005, 51 – 62.
20. FALCARIN, P., BALDI, M. AND MAZZOCHI, D. 2004. Software Tampering Detection using AOP and mobile code. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004, 1-6.

21. Software solutions for successful Management, On the necessity of management of information security.2002.[Online] Available: <http://www.noweco.com/wp_iso17799e.htm>,
22. FIEGE, L., ZEIDLER, A., BUCHMANN, A., DARMSTADT, T. U., KILIAN-KEHR, R. AND M¨UHL, G. 2004. Security Aspects in Publish/Subscribe Systems. In Third International Workshop on Distributed Event-Based Systems (DEBS 2004), Edinburgh, Scotland, UK, 24-25 May 2004, 1-7.
23. FOX, J. AND JURJENS, J. 2005. Introducing Security Aspects with Model Transformations. In *12th IEEE* International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), Greenbelt, Maryland, 4-7 April 2005, 543-549.
24. GAO, S., DENG, Y., HE, X., BEZNOSOV, K. AND COOPER, K. 2004. Applying Aspect-Orientation in Designing Security Systems: A Case Study. In The Sixteenth International Conference on Software Engineering and Knowledge Engineering, Alberta, Canada, 20-24 June 2004, 360-365.
25. GEORG, G., RAY, I. AND FRANCE, R. 2002. Using Aspects to Design a Secure System. In Eighth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'02), Greenbelt, Maryland, USA, 2-4 December 2002, 117-126.
26. GRUNDY, J. AND DING, G. 2002. Automatic Validation of Deployed J2EE Components Using Aspects. In 17th IEEE International Conference on Automated Software Engineering (ASE'02), Edinburgh, UK, 47-57.
27. HANKERSON, M. B.2003.Towards a Taxonomy of Aspect Oriented Programming, Masters Thesis. East Tennessee State University. Johnson City, Tennessee.
28. HASLE, H.2004.Aspect-Oriented Programming and A comparison between implementing JAAS with AOP and OOP, Masters Thesis. Norway, Gjøvik University College.
29. HAYLEY, C. B., LANEY, R. C. AND NUSEIBEH, B. 2004. Deriving Security Requirements from Crosscutting Threat Descriptions. In Proceedings of the 3rd international conference on Aspect-oriented software development, Lancaster, UK, 22-26 March 2004, 112-121.
30. HOUMB, S. H., GEORG, G., FRANCE, R. AND MATHESON, D. 2004. Using aspects to manage security risks in risk-driven development. In 3rd International Workshop on Critical Systems Development with UML, Lisbon, Portugal, 12 October 2004, 71-84.
31. HOUMB, S. H., GEORG, G., FRANCE, R. AND MATHESON, D. 2004. Using aspects to manage security risks in risk-driven development. In 3rd International Workshop on Critical Systems Development with UML, Lisbon, Portugal, 71-84.
32. HUANG, M., WANT, C. AND ZHANG, L. 2004. Toward a Reusable and Generic Security Aspect Library. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004, 1-6.
33. KAWAUCHI, K. AND MASUHARA, H. 2004. Dataflow Pointcut for Integrity Concerns. In AOSD 2004 Workshop on AOSD Technology for Application-level Security (AOSDSEC), Lancaster, UK, 22-26 March 2004, 1-6.
34. KICZALES, G., HILSDALE, E., HUGUNIN, J., KERSTEN, M. AND PALM, J. 2001.Getting Started with AspectJ. *Communications of the ACM,* 44(10)**,** 59-65.
35. KICZALES, G., HUGUNIN, J., KERSTEN, M., LAMPING, J., LOPES, C. AND W.G., G. 2000. Semantics-Based Crosscutting in AspectJ. In Workshop on Multi-Dimensional Separation of Concerns inSoftware Engineering (ICSE 2000).
36. KOCH, M. AND PAULS, K. 2005. Model-driven development of access control aspects. In *SICHERHEIT 2005*, Regensburg, Germany, 5-8 April 2005, 273-284.
37. KUMAR, A., SINGH, A. K. AND BABU, R. S. 2001.A security assurance framework for component based software development. *Informatica*, 25(4)**,** 509 - 515.
38. KUNTZE, N., RAUCH, T. AND SCHMIDT, A. U. 2005. Security for distributed web applications via aspect oriented security. In Conference Information Security South Africa, Sandton, South Africa, 29 June - 1 July 2005, 1-12.
39. LANDWEHR, C. E., BULL, A. R., MCDERMOTT, J. P. AND CHOI, W. S. 1994.A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys,* 26(3)**,** 211-254.
40. LANEY, C. R., VAN DER LINDEN, J. AND THOMAS, P. 2004. Evolution of Aspects for Legacy System Security. In AOSD workshop on dynamic aspects AOSDSEC'04, Lancaster, UK, 22-26 March 2004., 1-7.

41. LANEY, R. C., VAN DER LINDEN, J. AND THOMAS, P.2003.Evolving Legacy System Security Concerns Using Aspects. *Technical Report TR 2003/13.* Dept. Of Computing, The Open University, UK.
42. LI, X., NAEEM, N. A. AND KEMME, B. 2005. Fine-Granularity Access Control in 3-Tier Laboratory Information Systems. In Application Symposium (IDEAS'05) 9th International Database Engineering, 391-397.
43. MCGRAW, G. 2002.Managing Software Security Risks. *IEEE Computer*, 35(4)**,** 99-101.
44. MEADOWS, C. 1993. An outline of a taxonomy of computer security research and development. In *Proceedings on the 1992-1993 workshop on New security paradigms, Little Compton, Rode Island, August 1993, 33-35.*
45. MILLER, S. K. 2001.Aspect-Oriented Programming Takes Aim at Software Complexity. *Computer,* 34(4)**,** 18-21.
46. NAQVI, S. AND RIGUIDEL, M. 2005. Addressing Secure Access Challenges for Nomadic Grid: A Hospital Case Study'. In Grid Asia Conference 2005, Biopolis, Singapore, May 2-6, 2005.
47. PADAYACHEE, K. AND ELOFF, J. H. P. 2006. The Next Challenge: Aspect-Oriented Programming. In Proceedings of the Sixth IASTED International Conference on Modeling, Simulation and Optimization, Gaborone, Botswana, 11-13 September 2006, (Ed.)Nyongesa, H., ACTA Press, 304-307.
48. PALMER, D. 2002. Dynamic Aspect-Oriented Programming in an Untrusted Environment. In International Symposium on Distributed Objects and Applications (DOA), Irvine, California, 28-30 October 2002.
49. PAVLICH-MARISCAL, J., MICHEL, L. AND DEMURJIAN, S. 2005a. A Formal Enforcement Framework for Role-Based Access Control using Aspect-Oriented Programming. In Proceedings of ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (Models/UML 2005), Montego Bay, Jamaica, 2-7 October 2005, 537-552.
50. PAVLICH-MARISCAL, J. A., DEMURJIAN, S. A. AND MICHEL, L. D. 2005b. A Framework for Composable Security Definition, Assurance, and Enforcement. In ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica, 2-7 October 2005.
51. PAVLICH-MARISCAL, J. A., DOAN, T., MICHEL, L., S.A., D. AND T.C., T. 2005c.Role Slices: A Notation for RBAC Permission Assignment and Enforcement.
52. PFLEEGER, C. P. AND PFLEEGER, S. L. 2003. *Security in Computing.* Prentice Hall, Upper Saddle River, New Jersey,
53. RAMACHANDRAN, R., PEARCE, D. J. AND WELCH, I. 2006. AspectJ for Multilevel Security. In The 5th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Bonn, Germany, 2006, 21 March 2006.
54. REN, J., TAYLOR, R., DOURISH, P. AND REDMILES, D. 2005. Towards An Architectural Treatment of Software Security: A Connector-Centric Approach. SIGSOFT Softw. Eng. Notes, 30(4)**,** 1-7.
55. RITS, M. 2003. Component adaptability and security. In *DEA Réseau et Systèmes Distribués,* Université de Nice - Sophia Antipolis, 29 June 2003.
56. RITS, M., DE BOE, B. AND SCHAAD, A. 2005. XacT: A Bridge between Resource Management and Access Control in Multi-layered Applications. In Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications, St. Louis, Missouri, May 2005, 1-7.
57. ROBINSON, P., RITS, M. AND KILIAN-KEHR, R. 2004. An Aspect of Application Security Management. In AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004.
58. SHAH, V. AND HILL, F.2003a.An Aspect-Oriented Security Assurance Solution, Defence Advanced Research Projects Agency, [Online] Available: http://www.stormingmedia.us/50/5039/A503914.html.
59. SHAH, V. AND HILL, F. 2003b. An Aspect-Oriented Security Framework. In DARPA Information Survivability Conference and Exposition, Washington D.C, 22-24 April 2003, 143.
60. SHREYAS, D.2003.Software Engineering for Security: Towards Architecting Secure Software. Irvine, Information and Computer Science Department, University of California.
61. SLOWIKOWSKI, P. AND ZIELINSKI, K.2003. Comparison Study of Aspect-oriented and Container Managed Security. AGH University of Science and Technology.
62. SONG, E., REDDY, R., FRANCE, R., RAY, I., GEORG, G. AND ALEXANDER, R. 2005. Verifiable composition of access control and application features. In Proceedings of the tenth ACM symposium on Access control models and technologies, Stockholm, Sweden, 1-3 June 2005, 120 – 129.

63. VANHAUTE, B. AND DE WIN, B. 2001. AOP, Security and Genericity. In 1st Belgian AOSD Workshop, Vrije Universiteit Brussel, Brussels, Belgium, 8 November 2001.

64. VERHANNEMAN, T., PIESSENS, F., DE WIN, B., TRUYEN, E. AND JOOSEN, W. 2005. Implementing a modular access control service to support application-specific policies in CaesarJ. In AOMD '05: Proceedings of the 1st workshop on Aspect oriented middleware development, New York, NY, USA, ACM Press.

65. VIEGA, J., BLOCH, J. T. AND CHANDRA, P. 2001.Applying Aspect-Oriented Programming to Security. *Cutter IT Journal,* 14(2)**,** 31-39.

66. VIEGA, J. AND EVANS, D. 2000. Separation of concerns for security. In ICSE 2000 Workshop on Multi-Dimensional Separation of Concerns in Software Engineering, Limerick, Ireland, June 2000, (Eds.).Tarr, P., Harrison, W., Ossher, H., Finkelstein, A., Nuseibeh, B. ANDPerry, D., 126-129.

67. WALKER, R. J., BANIASSAD, E. L. A. AND MURPHY, G. C. 1999. An initial assessment of aspect-oriented programming. In Proceedings of the 21st international conference on Software engineering, Los Angeles, California, May 1999, 120-130.

68. WELCH, I. S. AND STROUD, R. J. 2003.Re-engineering Security as a Crosscutting Concern. The *Computer Journal,* 46(5)**,** 578 – 589.

69. XU, D. AND GOEL, V. 2005. An Aspect-Oriented Approach to Mobile Agent Access Control. In Proceedings of the International Conference on Information Technology: Coding and Computing(ITCC'05), Las Vegas, April 2005, IEEE Computer Society, Los Alamitos, CA, USA, 668-673.

70. XU, D. AND NYGARD, K. 2005. A threat-driven approach to modeling and verifying secure software. In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, Long Beach, CA, USA, ACM Press, New York, NY, USA, 342--346.

NOTES