

Implementing Application and Network Security using Aspect-oriented Programming

F.E. Tshivhase¹, H.S. Venter², J.H.P. Eloff³

¹tshivhasef@tuks.co.za, ²hventer@cs.up.ac.za, ³eloff@cs.up.ac.za

Information and Computer Security Architectures (ICSA) Research Group

Department of Computer Science

University of Pretoria

Pretoria, 0002

South Africa

Abstract – Network security is very important in modern day technology. Network technology is the most powerful tool that the world uses for communication. Security of software applications that are used to access information can influence security over the network. It is imperative for network designers to make sure that the network is as secure as possible. Software developers also need to embed or implement security in the software applications. Security can be easily implemented in the earlier stages of the software development process, however, some developers usually overlook security functionalities of a system because they believe it is time consuming. Network security issues are discussed to see how program security affects network security. We also focus on the techniques of using aspect-oriented programming to include security after development and, moreover, a way to enforce security awareness for developers during the software development process. The way security is implemented using aspect-oriented programming is also discussed. Aspect-oriented programming is a programming paradigm that aims at reducing the complexity of the program by allowing software developers to add security features after the initial software development phase is finished.

Keywords: Network security, aspect-oriented programming, security awareness, software development, code scattering and code tangling.

I. INTRODUCTION

Computer networks are essential for communication in the times we live in. This is due to the fact that people want flexibility. For instance, a driver wants to still have access to his or her email while he or she is in the car, a business analysts may still want access to some news update from the internet while on holiday in a game reserve somewhere. Networks have become sophisticated because many transactions such as e-commerce happen over the internet.

People are naturally sensitive to security. We always want to maintain our privacy and make sure that we are safe from security issues like disclosure of one's details while buying online. These issues bring forth a challenge to the network designers to make sure that everything is as secure as possible over the network. There is a close relationship between the security in a software application and network

security. A program used to access the network can also contribute on the overall security of the internet. This program could be an internet browser or any other software application. If there is some discrepancy with the software application then the security of the application is already compromised.

Having established the relationship of the role that security plays between the network and the software application, we show how designers of applications can deal with the security issues. The rest of the paper deals with implementing security during or after the initial software development phase, because we believe that, since network security is also affected by the application being used, it is necessary to start at the root of the problem: making sure that a software application is secured.

During software development, programmers strive to develop a software application that meets the user's requirement and that is secure and reliable. Some developers usually overlook the required security features in software applications because they tend to focus more on customer requirements. Although Aspect-oriented programming (AOP) makes it possible to patch in security after the initial development stage [7], it is just as important to introduce security in the earlier stages of the software development process, as to create a security awareness culture. We see security as an important feature of any system, even though it might not always be explicitly stated by a client as one of the requirements for the system.

In this paper, we explain how AOP can be used to increase security awareness amongst developers and some methods of enforcing security awareness among programmers are recommended on application and on network level. The next section of this paper delves into the background of AOP. Section III elaborates on how AOP can be used to implement security solely at the application level. Section IV presents a method that can be used to enforce security awareness among developers at the application level as well as on the network level. The last section concludes the paper and provides some future work.

II. THE BACKGROUND OF AOP

As mentioned in the introduction, there is a definite relationship on how application security can influence network security. The background in this section focuses on

AOP and how it can be used at an application level. Even though research shows that the use of object-oriented programming (OOP) has enhanced software development over the years, somehow developers still find it difficult to express a problem fully into a model that is completely modular and encapsulated [6]. AOP is a programming paradigm that aims at resolving some challenges that OOP could not address effectively. These challenges include code scattering and code tangling. Code scattering happens when the code required to fulfill one concern is spread over the classes required to fulfill another concern. Code tangling has to do with using a single method or class to implement multiple concerns [6]. A concern is the functionality or requirement that is essential in a system and is implemented in a code structure [6]. The aspect-oriented paradigm makes the code to be succinct and easy to reuse. AOP was invented at the Xerox Palo Alto Research Center by Gregor Kiczales and his colleagues [15].

AOP was also designed to encourage modularity of code. Modularity assists with eliminating code tangling and scattering. An application or a program with tangled code and scattered code is difficult to edit and debug. The methodology of breaking a problem into small modules of functionality has been used to solve problems until today [6]. AOP also addresses this problem by making use of aspects and it prevents programmers from making common mistakes like invoking a wrong method. An aspect is a programming construct that resides in its own file. It identifies a point of interest and operations to be applied. Point of interest is a specific part of the program that a specific action must be implemented and operations are the actions that are taken by the aspect code. An aspect is composed with other aspects that address security concerns in this case and it is independent of any programming language [13]. Figure 1 shows an example of an aspect. The aspect in Figure 1 makes it possible to replace all calls to the rand() function call with a secure version of the rand() function. The rand() function is advantageous because its output is completely reproducible. This means that the rand() function can be used again in another class because it is within an aspect that is in a separate file.

```

Aspect secure_random {
    int secure_rand(void) {
//Secure call to random is defined
here
    }
    funcCall<int rand(void)> {
    replace {
        secure_rand();
    }
    }
}

```

Figure 1: Example of an aspect [13]

AOP makes programming easy for developers since they do not have to always rewrite the same code for a specific behavior that has already been programmed. Users usually bring more additional concerns for the application during the development process and as a result other concerns can sometimes be easily overlooked.

One of the main purposes of AOP is that of specifying the structured transformation on a program. Structured transformation involves inserting or removing code at well-defined points [13]. AOP relies on the features of its host language, which is why the user does not have to learn so many new techniques. There are a number of benefits that results from using the AOP technique. The AOP approach improves performance since the methods or operations are more succinct and programmers spend less time not having to rewrite the same code. It is evident that AOP enables better encapsulation of different procedures and promotes future interoperation [10]. AOP has been around for about a decade now. There have been some security implementations using AOP as will be discussed in the next section.

III. HOW SECURITY IS BEING IMPLEMENTED USING AOP

Security should be, by default, the de facto standard in any application no matter the language used. For the sake of reuse, maintainability and clarity, security-related elements (pieces of the code) should be abstracted properly in a program. The AOP technique allows programmers to separate security concerns from the code. This enables programmers to just focus on developing the main application and security experts to specify the security properties that need to be present in an application [13]. It has been shown that developers are not very concerned about developing software applications that need to be secured [16].

No modification of the application is required for sources to introduce security when it comes to Aspect-oriented security. Aspect-oriented security is highly flexible and extensible [8]. This is due to the ability of AOP to weave or intertwine the concerns by making use of the aspects. By making use of AOP there are mechanisms of software tempering detection that have been implemented in applications running on un-trusted hosts [5]. An aspect-oriented program was used to do self-checking. Self-checking is a process where a program checks itself to verify that it has not been modified [8].

According to findings from the experiment that was conducted by Bostrom [2]; database encryption can be added after the completion of the system by using AOP. Most security-typed languages have a problem of exposing programmers to the data that is confidential and the data that is public. Full information might not be available during functional design of the system [12]. The other problem is that the programmer is supposed to consider security together with functional requirements.

Software development in AOP is applicable to all main pillars of security in computing. These main pillars include authentication, access control, integrity, and it also supports administration and monitoring disciplines [1]. According to Laney et al. [7] security aspects can be used to modularize access control and authentication [4, 9, 11]. Authentication is the process of establishing and verifying the claimed identity of the user. Access control is the prevention of unauthorized use of a resource and it also includes the use of a resource in an unauthorized manner.

Although AOP makes it possible for the programmer to incorporate security into a finished program, it is more effective to include security during the development stage. In the next section, we suggest the methodology that can be used to enforce awareness to programmers so that they can include security concerns even while developing the software application.

IV. THE METHODOLOGIES FOR ENFORCING SECURITY AWARENESS

Computer networks serves to be a platform where applications come together, interact and merge. These applications act as a middle layer between network users and computers. People use software applications i.e. internet web browsers to access the network. It is therefore important to view network security from the application level as well.

We focus on two approaches of implementing security using AOP, which include security at the application level and security at the network level. Figure 2 shows how the two approaches can still be further divided into other approaches. Security at the application level can be implemented in two ways. A security checker (explained in detail in subsection A) can be used to implement security from the initial stage of development. Security can also be implemented by just adding a new security module into an existing application using AOP. The idea of implementing security during the development phase will be explained in subsection A.

The network is populated by applications that are developed by different designers and developers. Security at network level can also be implemented in two ways. The method of using a security checker and that of adding a new security module afterwards are applied at the network level because the network comprise of applications that are already running

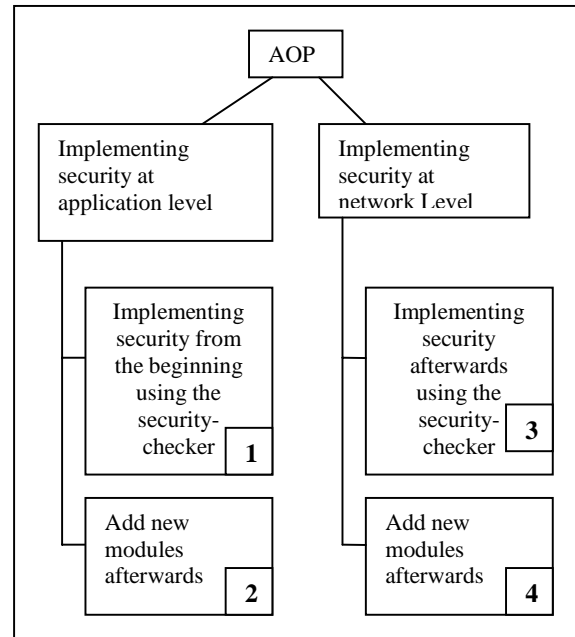


Figure 2: The breakdown of implementing security using AOP

The following subsection explains the idea of the security checker.

A. Implementing security from the beginning using a security checker

This subsection deals with implementing security from the beginning of the application development process by using a security-checker class as shown by block 1 in Figure 2. Developers often know beforehand that security is a requirement for a program. However, they often jump right into the development of the functional requirements of a program, only to return after the initial development process is complete and then attempt to implement security requirements. This approach often makes it very difficult to implement security requirements afterwards. There are certain tools that can be used to evaluate the vulnerability status of a system. These tools, however, do not suggest how developers should design and implement the system in a more security-conscious way to avoid the same problem recurring [13].

Despite enormous research efforts and debates, some improvement to security in software applications is still much needed. Due to the fact that programmers may ignore security during the development stage, one can try and conquer the problem by introducing a security class that all programs will inherit from. The class comes with security embedded and it prompts the programmer to include the necessary security features in the program while he/she will be busy programming. We will refer to this class as a security-checker class. This idea was born after looking at numerous security awareness research done in the past [3, 5, 6, 8, 13, 14]. The specific example that we will use to demonstrate the idea is that of a login program. If the programmer is developing a system that involves that kind

of security (i.e. the logon method), the security class will prompt the user in a form of a compile error to include the logon method after attempting to compile that program. The program might not run until the necessary method is implemented.

B. Implementing security afterwards

This subsection deals with implementing security after the application development process as shown in blocks 2, 3 and 4 of Figure 2. AOP enables a programmer to simply plug in the class and continue enhancing the application by rather dwelling on the functional concerns. The security concern at hand will be implemented automatically and eventually it will become intuitive to the programmer to incorporate security concerns. This approach involves having a module that is coded just specifically for implementing new security modules afterwards as shown in blocks 2 and 3 in Figure 2. The module will be represented in the form of an aspect using AOP.

Figure 3 shows an example of the aspect code that is used in the weaving of the program and the security class. The `final_class` is the main class that is implementing the security class.

```
Define aspect final_class {
    Class_security s_Object =
        new class_security();
    when calling set*(taking one parameter) {
        s_Object.log("Calling the security method");
    }
}
```

Figure 3: Code for the aspect used for weaving

When the security class is called, it will first execute the defined method in the aspect language. It checks for any method that begins with the word `set`. In the login/password example, the `setPassword` method will be called and before it actually executes the statement "calling the security method" will be displayed as shown in Figure 3. This will help in reminding the programmer on where the program is at that current state. In this example, AOP aids in forcing every call that matches the criteria that has been set on the aspect to include the security class.

The methodology that we are introducing here is to improve the awareness of programmers as far as security is concerned. We believe that making the programmer more aware of security will make the issue more intuitive and practical for the programmers. With this idea, we envisage that, programmers will intuitively include security without having to be reminded.

The idea of adding new security modules will be addressed in the future work where an actual prototype of all the approaches will be demonstrated. The following section will outline the work that was covered in this paper and remark on the future work.

V. CONCLUSION

The impact that security on software applications have on the network security makes it to be imperative for the software developers and network designers to make sure that the applications are as stable and secure as possible. Poor security in a system can cause vulnerabilities to arise in the system. Intruders may easily hack into the system by exploiting vulnerabilities. This suggests that programmers must make sure that security is implemented in the software applications that they develop. Programmers should therefore include security features into a program during the development process.

AOP minimizes the effort and it saves the user or the developer a great deal of time. Having looked at the security of the software programs, it is clear that AOP makes a huge impact in the modularity process of programming. AOP produces the notion that a developer can simply plug in the security aspect even when the whole software development cycle has completed. The purpose of the paper is based on a way of making programmers more conscious about incorporating security in their programs. As it was suggested in Section IV, programmers need to be reminded by means of having a special class that will check the security implementation in the program that is being developed. If no security has been implemented, a compile error should occur.

As this idea is still at a conceptual stage, we aim to do some future work in developing an actual prototype for demonstrating the idea.

REFERENCES

- [1] Bodkin, R. 2004. Enterprise Security Aspects, *AOSD 2004 Workshop, Lancaster, UK, 23 March 2004*. Available online: <http://www.cs.kuleuven.ac.be/~ditrinet/events/aosdsec/papers.html>
- [2] Bostrom, G. 2004. Database encryption as an Aspect, *AOSD 2004 Workshop, Lancaster, UK, 23 March 2004*. Available online: <http://www.cs.kuleuven.ac.be/~ditrinet/events/aosdsec/papers.html>
- [3] De Win, B., Vanhaute, B. & De Decker, B. 2002. Security through aspect oriented programming. In De Decker B., Piessens F., Smits J., & Van Herreweghen E. eds *Advances in Network and Distributed Systems Security. IFIP TC11 WG11.4 First Working conference on Network Security Leuven, Belgium 26-27 November 2001*: 125 – 138.
- [4] De Win, B., Joosen, W. and Piessens, F. 2003. AOSD and Security: a practical assessment. In Workshop on Software Engineering Properties of Languages for Aspect Technologies (SPLAT03): Boston. Available online: http://www.daimi.au.dk/~eernt/splat03/papers/Bar_t_De_Win.pdf

- [5] Falcarin, P., Baldi, M. and Mazzochi, D. 2004. Software Tampering Detection using ASPECT-ORIENTED PROGRAMMING and mobile code, AOSD 2004 Workshop, Lancaster, UK, 23 March 2004. Available online: <http://www.cs.kuleuven.ac.be/~ditrinet/events/aosdsec/papers.html>
- [6] Gradecki, J.D., Lesiecki, N. Gradecki, J 2003 Mastering AspectJ: Aspect-Oriented Programming in Java, *Chapter1*. Available online: http://media.wiley.com/product_data/excerpt/44/04714310/0471431044.pdf
- [7] Laney, R.C., van der Linden, J. Thoma, P. 2003. Evolving Legacy System Security Concerns Using Aspects, *Technical Report TR 2003/13*, Department of Computing, The Open University, UK, 2003. Available online: <http://chilli.open.ac.uk/tn.index.php/2003/200313>
- [8] Padayachee, K. 2005 Aspect-oriented Programming and Security, ICSA, University of Pretoria.
- [9] Shah, V. and Hill, F. 2003. An aspect-oriented security framework, *In Proceedings DARPA Information Survivability Conference and Exposition Washington, DC, USA, 22-24 April 2003*.
- [10] Shukla, D., Fell, S. and Sells, C. 2002. Aspect-Oriented Programming Enables Better Code Encapsulation and Reuse, *In MSDN Magazine March 2002*.
- [11] Slowikowski, P. and Zielinski, K. 2003. Comparison Study of Aspect-Oriented and Container Managed Security, AAOS2003: Analysis of Aspect-Oriented Software. Workshop held in conjunction with ECOOP 2003 Darmstadt, Germany.
- [12] Tse, S. & Zdance, S. 2004. Run-time Principals in Information-flow Type Systems, *In Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P 2004) Bekerly, California, 09 – 12 May 2004* : 179-193.
- [13] Viega, J., Bloch, J.T., Chandra, P. 2001 Cutter IT Journal Vol14, No. 2, *Applying Aspect-Oriented Programming to security*.
- [14] Viega J. and Voas, J. 2000 Can Aspect-Oriented Programming Lead to More Reliable Software, IEEE Software.
- [15] Kiczales, G. Lamping, J. Mendhekar, A. Maeda, C. Lopes, C.V. Loingtier, J. Irwin, J. 1997 *Aspect-Oriented Programming: Xerox Palo Alto Research Cente. European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241.1997*.
- [16] Robinson, P., Rits, M., Kilian-Kehr, R. 2004. *An Aspect of Application Security Management, AOSD 2004 Workshop, Lancaster, UK, 23 March 2004*. Available online: <http://www.cs.kuleuven.ac.be/~ditrinet/events/aosdsec/papers.html>

Fulufhelo Emmanuel Tshivhase born in Limpopo, South Africa. Obtained his BSc. degree in 2004 at the University of Pretoria, South Africa. He is currently studying for his Honours degree in Computer Science at the University of Pretoria, South Africa.

