The Secure Pipes and Filters pattern

Eduardo B. Fernandez
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
ed@cse.fau.edu

Jorge L. Ortega-Arjona
Departamento de Matemáticas
Facultad de Ciencias, UNAM
México
jloa@ciencias.unam.mx

Abstract—**Many applications process or transform a stream of data. Such applications are organized as a sequence of different stages, which may be independent enough to be simultaneously carried out. The original Pipes and Filters pattern and the Parallel Pipes and Filters pattern describe those actions. We present here the Secure Pipes and Filters pattern as a secure version of the original patterns, which contains a minimal set of security mechanisms to provide a set of basic security functions. The Secure Pipes and Filters pattern includes ways to add security controls at each stage of processing, controlling that only predefined operations are applied to data streams, as well as securing data movement.**

Keywords-**Security; Pipes and Filters pattern; security controls**

I. Introduction

Many applications process or transform a stream of data. Such applications are organized as a sequence of different stages, which may be independent enough to be simultaneously carried out. The value of this organization is due to several reasons: every component performs specialized functions over the data stream, the global architecture requires this flow, or the whole system performs its functionality in a more efficient and flexible way. Every time the data reaches a different stage, different functions may be applied at that stage. In its most common form, the filtering just applies automatic, predefined data transformations. Examples of this type include compilers [4] and document generation [20]. In some applications, however, the pipeline implements a business process where the data transformations may include several operations applied interactively by people. An example of the latter case is a pipeline of workers processing tax returns.

This organization of the process into stages has been converted into a pattern: the Pipes and Filters pattern [4], [16]. However, the descriptions provided for this pattern take into consideration only its functional properties, and a few non-functional properties such as its potential for improving performance [16]. Other non-functional properties, such as security, are not considered. [13] indicates that each filter needs its own security but doesn't provide details. As indicated, the use of Pipes and Filters is capable of fitting either a human-interactive system or an automatic processing environment. Of course, between them there is a great difference of performance, since humans require more time than processors. But the principle is the same for both: there are components (filters) where data is modified with certain objectives, and communications (pipes) that allow the flow of data between components. Moreover, security is an issue that can be added in both situations: when using a human-interactive application, we need to make sure that the person interacting with the filter is the person who has authorization to "process" it. On the other hand, in an automatic processing environment, we need to control before the stream passes through the pipeline who can define the operations to be applied to the stream. We can also need security functions such as logging and authentication and the transfer of data between stages may also need to be protected.

We have proposed a methodology that helps developers build secure systems through the use of patterns [8]. Our discussion here is independent of this methodology although this paper is a consequence of this work. Patterns provide solutions to recurrent problems and are abstractions of best practices. We see the use of patterns as a fundamental way to incorporate security principles in the design process even by people having little experience with security practices. Security patterns are relatively new and starting to be accepted by industry because they are useful to guide the security design of systems by providing generic solutions that can prevent a variety of attacks [18]. Our methodology starts from the analysis of use cases and their threats and obtains a conceptual model where security mechanisms are embedded in the form of pattern instantiations. We them map this model to the software artifacts of the design stage. We have shown how patterns allow us to define secure architectures for middleware systems [9]. We build a secure architecture as a composition of functional (insecure) patterns with patterns that provide specific security functions that can control some threats. To use our approach we need a catalog of relevant patterns. In particular, we are writing secure versions of the distribution patterns in [4]. Each pattern is made of a core distribution pattern, e.g. Broker, to which a minimal set of security patterns is added to provide a secure system. An important pattern in distributed systems is the Secure Pipes and Filters pattern and we present here a secure version, the Secure Pipes and Filters. This pattern is oriented to designers of pipeline systems, who need to plan where they should apply security controls, and to users of such systems, who need to configure these controls. The pattern considers the worst case, where we need security controls in each unit.

Section II presents the pattern following the template used in [4]. We do not repeat here in detail the functional features of this pattern, we only show its security properties. Section III provides some conclusions and future work.

II. The Secure Pipes and Filters pattern

A. Example

ArtisticRenderings is a company that prepares brochures and reports for marketing real estate, stocks, and all kind of products. To prepare each brochure we need a product specialist, a graphic designer, and an artist. To insert information from databases, e.g. sales statistics, we need some IT people. The whole process is under the control of a supervisor. Each person has its own interface and once they complete their jobs their inputs will be applied in sequence to the stream of documents. However, some documents are sensitive and we need to control who makes the changes or a disgruntled employee might introduce incorrect contents.

B. Context

We consider Pipes and Filters software or other processing systems which are used to process data streams. Some of them may be parallel, attempting to improve the process performance. The execution platform for this kind of system is frequently a distributed one, whose components may require a certain level of security for processing the stream of data. "Parallel" here means that several components (whether human or automatic) act simultaneously. Even a human-interactive Pipes and Filters aims to improve performance. Even when we don't have significant performance improvement this architecture may be valuable for flexibility reasons or to have a systematic, well-structured process.

C. Problem

How to keep an acceptable level of security among the components of the whole pipeline system in the presence of possible attacks?

The essence of the Pipes and Filters pattern is that every time data reaches a different stage, different functions are applied on it, and in a secure version these actions should be controlled. In this kind of system, we may also need the flexibility to reorder the steps of the process or change the processing steps. In the example above, a new person may be assigned to the workflow to perform additional functions on the documents which may require adding an extra step. How do we control the actions to be performed in a data pipeline? Additionally, the data may be moved along the pipeline using insecure channels and the users defining the data transformations may be remote.

This problem requires considering the following forces:

- The system may need to control in each stage of processing who can do what (what operations can be applied) with the data in the pipeline. This may be necessary in both automatic and interactive pipelines. Otherwise, employees might introduce illegal content or filter wanted aspects.
- We might require that before the data is accepted by the next stage, the previous stage or the message carrying the data must be authenticated. Otherwise, an impostor might send the data to be processed.
- Before sending the data in the pipes we may need to hide it to prevent eavesdropping.
- Due to regulatory constraints, work changes, or efficiency, some documents may need extra stages or skip stages. We need to be able to reconfigure the number or order of the steps. This reconfiguration should be controlled or a user might skip needed stages or add unintended stages.
- We should keep track of any actions applied on the data in cases where legal documents or regulation-compliance is involved.
- The security controls should not affect the functional use of the system.
- The security controls should not significantly affect performance.

D. Solution

The Secure Pipes and Filters pattern provides a secure way to process data into different stages or steps, by adding in each of them basic security mechanisms (as instances of security patterns) to provide authentication, authorization, information hiding, and logging. Because the functions to be performed in each stage depend on persons doing specific tasks, we use a Role-Based Access Control (RBAC) model [7] to describe their required rights. A RBAC model assigns rights to roles to access data or resources in specific ways. Individual users may belong to one or more roles.

E. Structure

Figure 1 shows a class model for the RBAC pattern [7], [18]. In this model, Users are members of Roles and Rights are assigned to roles. A right defines the access type that can be applied by a role to a Protection Object.

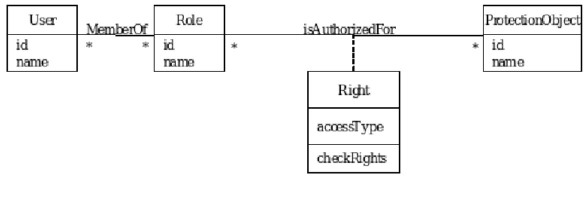

Figure 1. Class diagram for the Role-Based Access Control pattern

Pattern instances corresponding to security mechanisms have been added to the Pipes and Filters pattern in Figure 2. Since we are considering a set of stages the pattern is clearer by showing an object diagram (describing three typical stages) and not a class diagram. The subsystems named `Authenticator` are instances of the Authenticator pattern [18] and allow each Filter to authenticate the sender of the data it is receiving. `Log i` indicates instances of the Logging pattern, used to keep track of any accesses to the data. Objects with the stereotype `<<role>>` and `Right` are instances of the RBAC pattern. For example, `Role1` has the right to apply operations `op1` and `op2` to the data in `Filter i`. The Reference Monitor subsystem indicates the enforcement of the authorization rights defined by the RBAC instances [18]. We show the Reference Monitor as a shared resource and the Authenticators as individual for each stage; their actual distribution depends on the distribution architecture of the complete system. In order to control the reconfiguration of the stages, the RBAC pattern is also applied to the pipeline structure so that only someone with an administrator role (`Role3`) could perform any changes to it.

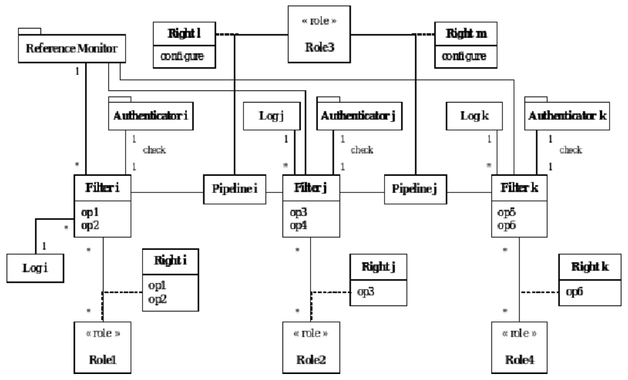

Figure 2. Object diagram for the Secure Pipes and Filters pattern

The RBAC pattern provides the option to abstract different roles within the data flow. It could be possible that we need to work with individual subjects instead of roles; in this case implementing the Access Matrix should be a better approach [7]. The link between stages could be subject to attacks, and optional operations of encryption and decryption could be implemented in each filter, as well as digital signatures in each data message (not shown in the diagram).

F. Dynamics

Figure 3 shows the use case where a subject with a specific role tries to execute an operation, `op3`, on a document. The Reference Monitor checks if its role allows the operation and if true it reads data from the input pipe, `Pipe i`, to the filter where `op3` is applied. After the operation the data is moved to the next pipe, `Pipe j`.

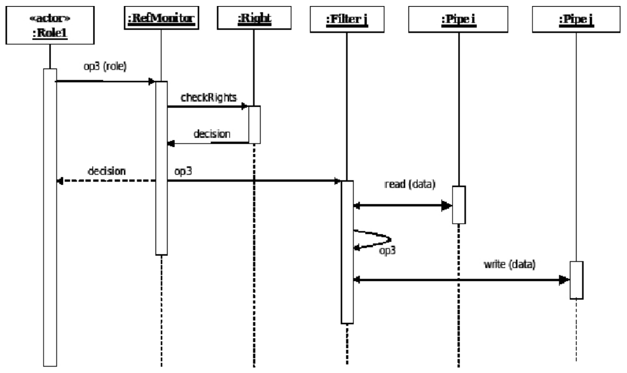

Figure 3. Sequence diagram for use case to apply an operation on a data stream

G. Implementation

We follow the steps suggested in [4] and indicate where security is needed:

1) Divide the application into a sequence of stages. As we discussed in [8], who should have access to which operations or results from each stage should be defined in the conceptual model. When the application is divided into stages we need to define how the rights in the complete model are reflected in each stage.
2) Define the data format to be passed along each pipe. This aspect has no effect on security.
3) Perform threat enumeration [8]and risk analysis. This is necessary to decide about what security mechanisms to add in each stage.
4) Decide how to implement each pipe connection. Aspects such as active or passive components, push or pull movement of data are defined at this moment. At this moment we have to decide if we use or not authentication between filters and if we do, what type of authentication. For communications within the same physical building, filter authentication may not be required although user authentication to the system is always needed.
5) Design and implement the filters. Each filter enforces the rights defined in the first implementation step and must have a Reference Monitor and a way to access the authorization rules. In distributed systems, one needs to decide where these rules should be stored. Filters also implement logging as well as encryption/decryption.
6) Design error handling. From the security side this implies handling security violations. This handling is application-dependent and no general policy is possible.
7) Set up the processing pipeline. The initial configuration as well as changes to the configuration must be restricted only to administrators.

H. Example Resolved

We implemented RBAC in the Pipes and Filters of the example. Now people making changes to the documents need to be authorized before they can make any change. The operations they can apply depend on their roles with respect to the application. Logging protects the company in case they need to show that they comply with regulations and they can track who made a specific change to a given document. Authentication is needed to apply authorization and maybe in between stages if necessary.

I. Known Uses

Microsoft's BizTalk Server 2004 [1]. BizTalk Server 2004 can implement Pipes and Filters. In addition to security features that are provided by the transport, such as encryption when using HTTPS, BizTalk Server 2004 provides security at the message level. BizTalk Server 2004 can receive decrypted messages and validate digital signatures that are attached to these messages. Similarly, it can encrypt messages and attach digital signatures to messages before sending them.

Cocoon [6]. The Apache Cocoon is a web development framework using components. It can be used to build XML pipelines, in which security restrictions can be added.

The tax offices of some countries implement a human pipeline to process tax returns. Workers may check different aspects of a tax return either manually or using computers and need to be authorized to do this.

[20] discusses the use of XML pipelines for document preparation, including stages for adding content, formatting, and personalization. What is done in each stage can be controlled.

Pipelines are common for data reduction when there are large volumes of data. [19] discusses a data reduction pipeline for spectroscopic data, where different transformations by different researchers are applied at each stage. What is done at each stage is controlled according to the functions of the researchers.

J. Consequences

The use of this pattern yields the following benefits:

- We have applied the principle of "defense in depth", defining a coherent set of security mechanisms that define a secure core for this application. In some cases, specific mechanisms can be left out, being careful about security consistency; for example, authorization requires authentication. In other cases, more security controls may be needed to prevent for example, conflicts of interest.
- We can assign privileges according to the functions needed at each stage of processing and the roles of those performing the functions. The use of operations over the data, can be restricted according to the rules of either RBAC or Access Matrix models.
- Each Filter stage can authenticate its users before they are authorized to perform specific functions and can authenticate the Filter sending data to it. Authentication is necessary if we apply authorization at each filter.
- The use of encryption between stages is possible, adding the possibilities of secure messages (prevents eavesdropping) and digital signatures (confirm the origin of a message).
- The Administrator role can control the reconfiguration of stages to accommodate changes in the process.
- Logging can be performed in each stage to keep track of any accesses and changes to the data. We can prove this way we have followed regulations, we can prosecute illegal actions, and we can improve the system if it did not prevent an attack.
- The security restrictions are transparent to the users (while they do not attempt illegal actions).

Applying this pattern imposes the following liabilities:

- The general performance of the system worsens due to the overhead of the security checks. With careful implementations of these functions the loss in performance should be small. For example, encryption/decryption takes time and should be used only when needed; access control to the filters happens only when a new type of data is being analyzed. In parallel pipelines the performance loss can be further reduced by performing some security functions in parallel with normal functions.
- The system is more complex, due to the extra services that we have added.

K. See Also

- [4] and [14] present the basic Pipes and Filters, without security controls.
- The Access Matrix, RBAC, and Authenticator patterns can be used to secure the stages [18].
- The Secure Channel pattern can be used to secure the communications channels [2].

III. Conclusion

The use of data pipelines or streams is frequent in software design. Following the principle that security must be applied in all stages of the software development, the designer should include security aspects in any application using pipelines. As mentioned earlier, this pattern may become part of a catalog of security patterns. Combined with other similar patterns, it gives a designer a choice of possibilities when building the middleware of a complex system [9]. A designer with little security experience can use a secure pipeline that comes with a basic set of security services.

We just presented a Secure Blackboard pattern [17]. For completion, we still need Secure Adapter, Secure MVC, and Secure Component; these are now in preparation. We also need more patterns for web services security. Implementing these patterns together in a real application would be another useful direction that would confirm the value of using patterns.

References

[1] *Implementing Pipes and Filters with BizTalk Server 2004* http://msdn2.microsoft.com/en-us/library/ms978668.aspx

[2] A. Braga, C. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic object-oriented software", Chapter 16 in *Pattern Languages of Program Design 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in *Procs. of PLoP'98*, http://jerry.cs.uiuc.edu/ plop/plop98/final_submissions/

[3] W. Brogden, "Using XML pipelines" Part 1: Sept. 6, 2006, Part 2: Oct. 3, 2006, http://searchwebservices.techtarget.com/home/0,289692,sid26,00.html

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns* Volume 1, West Sussex, England: John Wiley & Sons, 1996.

[5] F. Buschmann, K. Henney, and D.C. Schmidt, *Pattern-Oriented Software Architecture, Vol. 4: A Pattern Language for Distributed Computing* J. Wiley & Sons, UK, 2007.

[6] *The Apache Cocoon project*. http://cocoon.apache.org

[7] E. B. Fernandez and R. Pan, "A Pattern Language for security models", *Procs. of the 8th Annual Conference on Pattern Languages of Programs (PLoP 2001)* 11-15 September 2001, Allerton Park Monticello, Illinois, USA, 2001. http://jerry.cs.uiuc.edu/ plop/plop2001/accepted_submissions

[8] E. B. Fernandez, M.M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A methodology to develop secure systems using patterns", Chapter 5 in *Integrating security and software engineering: Advances and future vision* H. Mouratidis and P. Giorgini (Eds.), IDEA Press, 2006, 107-126.

[9] E. B. Fernandez and M. M. Larrondo-Petrie, "Developing secure architectures for middleware systems", *Procs. of CLEI 2006. (XXXII Conferencia Latinoamericana de Informática)*.

[10] E.B.Fernandez, N.A.Delessy, and M.M. Larrondo-Petrie, "Patterns for web services security", in *Best Practices and Methodologies in Service-Oriented Architectures* L. A. Skar and A.A.Bjerkestrand (Eds.), 29-39, part of OOPSLA 2006, the 21st Int. Conf. on Object-Oriented Programming, Systems, Languages, and Applications, Portland,OR, ACM, October 22-26.

[11] E. B. Fernandez, D. L. laRed M., J. Forneron, V. E. Uribe, and G. Rodriguez G. "A secure analysis pattern for handling legal cases", *Procs. of the 6th Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP'2007)* 178-187. http://sugarloafplop.dsc.upe.br/AnaisSugar2007_WEB.pdf

[12] E.B.Fernandez, M. Fonoage, M. VanHilst, and M. Marta, "The secure three-tier architecture", *Procs. of the Second Workshop on Engineering Complex Distributed Systems (ECDS 2008)* Barcelona, Spain, March 4-7, 2008. 555-560.

[13] N. Harrison and P. Avgeriou, "Leveraging Architecture Patterns to Satisfy Quality Attributes", *First European Conference on Software Architecture* Madrid, Spain, September 24-26, 2007, Springer Lecture Notes in Computer Science.

[14] Microsoft, "Pipes and filters" http://msdn2.microsoft.com/en-us/library/ms978599.aspx

[15] P. Morrison and E.B.Fernandez, "Securing the Broker pattern", *Procs. of the 11th European Conf. on Pattern Languages of Programs (EuroPLoP 2006)* http://www.hillside.net/europlop/

[16] J.L. Ortega-Arjona, "The Pipes and Filters Pattern. A Functional Parallelism Architectural Pattern for Parallel Programming." *Procs.of the 10th European Conference on Pattern Languages of Programming and Computing (EuroPLoP 2005)* http://www.matematicas.unam.mx/jloa/publicaciones/EuroPLoP2005.pdf

[17] J. L. Ortega-Arjona and E.B.Fernandez, "The Secure Blackboard pattern". *Procs. 15th Int.Conference on Pattern Languages of Programs (PLoP 2008).*

[18] M. Schumacher, E.B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering* Wiley 2006.

[19] M. Scodeggio et al., "The VVDS data-reduction pipeline: Introducing VIPGI, the VIMOS interactive pipeline and graphical interface", *Pubs. of the Astronomical Society of the Pacific* Vol. 117, November 2005, 1284-1295.

[20] J. Tennison, "Managing complex document generation through pipelining" http://idealliance.org/proceedings/xtech05/papers/04-03-01/