Computer Standards & Interfaces 36 (2014) 748–758

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Enterprise security pattern: A model-driven architecture instance

Santiago Moral-García [a,b,*], Santiago Moral-Rubio [c], Eduardo B. Fernández [d], Eduardo Fernández-Medina [e]

[a] Produban, Santander Bank, Boston, MA, USA
[b] Kybele Research Group, Dept. of Computer Languages and Systems II, Rey Juan Carlos University, Madrid, Spain
[c] BBVA Group, Madrid, Spain
[d] Secure Systems Research Group, Dept. of Comp. and Elect. Eng. and Comp. Science, Florida Atlantic University, Boca Raton, FL, USA
[e] GSyA Research Group, Dept. of Information Technologies and Systems, University of Castilla-La Mancha, Ciudad Real, Spain

ARTICLE INFO

Available online 30 December 2013

Keywords:
Secure cloud computing
Model driven architecture
Enterprise security architecture
Security pattern
Enterprise security pattern

ABSTRACT

To secure their information assets, organizations should seek support from enterprise security architectures. Security patterns are a good way to build and test new security mechanisms, but they have some limitations related to their usability. In previous work, we defined a new type of security pattern called *Enterprise Security Pattern*. The main objective of these patterns is to provide an instance of model-driven architecture, which offers a solution to recurring problems that have to do with information systems security. In recent years, the hiring of Software as a Service (SaaS) from cloud providers has become very popular. There seem to be many advantages of using these services, but organizations need to be aware of a variety of threats, as well as being prepared to handle them. In another work undertaken previously, we defined an enterprise security pattern called *Secure Software as a Service* (*Secure SaaS*), which the organizations could apply to protect their information assets when using SaaS. In this paper, we present different instances of the solution models of the enterprise security pattern *Secure SaaS*, aiming to verify the risks that an organization would assume if each of the instances were deployed. With this approach, we intend to show how the design decisions adopted when performing the transformations between the solution models can have a direct impact on the security provided by the pattern.

1. Introduction

Technological advances are currently improving many aspects related to the development and design of information systems, thus entailing an increase in the complexity of enterprise security architectures. This in turn brings about a rise in the number of attacks [15]. The biggest problem that we have encountered in the design of enterprise security architectures is that different information security engineers provide a variety of solutions to the same problem. Noting the large number of security engineers who may work for the same organization, it is seen to be necessary to find a carefully thought and proven set of security guidelines to assist engineers when designing enterprise security architectures.

An approach which is considered promising for this purpose is the use of security patterns [20]. Security patterns provide guidelines to support the construction and evaluation of new security mechanisms [5,17]. The use of security patterns helps to incorporate security principles when building secure systems [16]. However, they have some limitations:

- They are small units of defense. They can only handle one (or a few) threat. Considering the number of threats that can affect current information systems, a security designer should select an extensive set of security patterns when building secure systems.
- It is necessary to have different versions of the same pattern, one for each architectural level. As the building of secure systems needs an extensive set of security patterns, this fact increases the complexity when a security designer is trying to select a pattern.
- Several instantiations of a pattern may have common aspects, but the designer has to recognize them; failing to do so may cause unnecessary redundancies.

In previous work [13], and bearing these limitations in mind, we defined a new type of security pattern called *Enterprise Security Pattern*. The main objective of these patterns is to provide a top-down strategy based on models for defining enterprise security architectures across different levels of abstraction, including their technological implementation. Organizations would be able to use enterprise security patterns to select a global security strategy for a specific business process. This provides their designers with a set of security guidelines thereby standardizing the design and building of the enterprise security architecture for that process. In addition, security engineers could, on one hand, manage security elements included in the different abstraction models separately. On the other hand, they would be enabled to perform automatic transformations between them. This fact would make it easier for the designer to select and tailor the enterprise security patterns when s/he is building enterprise security architectures.

* Corresponding author at: 2 Morrisey Boulevard, DORCHESTER, 02125 MA, USA.
E-mail addresses: smoralga@produban.us, santiago.moral@urjc.es (S. Moral-García), santiago.moral@bbva.com (S. Moral-Rubio), ed@cse.fau.edu (E.B. Fernández), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina).

0920-5489/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.csi.2013.12.009

The practice of outsourcing business functions has been around for decades. In recent years, it has become very popular as an online software service [4]. Online software delivery is now conceived and defined as Software as a Service (SaaS). SaaS focuses on separating the *possession* and *ownership* of software from its *use* [19]. The advantages of using SaaS seem to be numerous, because this online service model may prove cheaper than owning and maintaining an in-house IT system [11]. Companies expect to save money on support and upgrade costs, IT infrastructure, IT personnel, and implementation. However, this new environment brings some threats that organizations need to deal with. In another previous piece of work [14], we defined an enterprise security pattern called *Secure Software as a Service* (*Secure SaaS*), the aim of which was to provide proven and reliable enterprise security architecture that assures the information assets when organizations decide to outsource their applications.

In this paper, which has evolved from the work mentioned previously [14], we present different instances of the solution models of the enterprise security pattern *Secure SaaS*, aiming to find out the risks which an organization would incur if each of the instances were deployed. With this approach, we intend to show how the design decisions adopted when performing the transformations between the solution models can have a direct impact on the security provided by the pattern.

The remainder of this paper is organized as follows. Section 2 provides a description of the concept of enterprise security patterns, including their context and solution model. Section 3 presents the enterprise security pattern *Secure Software as a Service* (*SaaS*). Section 4 shows different transformations for each of the solution models. Finally, Section 5 presents some conclusions and future work.

2. Enterprise security patterns

The biggest problem that we have encountered when designing enterprise security architectures is that different information security engineers provide different solutions to the same problem, because each one has his or her own guidelines, or follows his or her own criteria. In addition, the number of iterations when developing the architecture and the reusability level between one design and the next one depends on the experience and quality of the engineer, rather than on the method used.

One of the basic principles of information security is that there are no formal mechanisms to verify the robustness of enterprise security architectures. This means that the number of valid architectures is low and can only be validated by taking account of the time that they have been operating in the industry without being violated. Similarly, organizations should use only enterprise security architectures that have been proved on the basis of their history of absence of incidents. Given these problems, it becomes necessary to find a careful thought and proven set of security guidelines for designing enterprise security architectures; it needs to be a set that is based on the historical validation of the mechanisms used. This set of security guidelines would help to reduce significantly the risk of designing architectures which are apparently useful, but which actually have security weaknesses.

Enterprise security patterns offer solutions to recurring problems related to information systems security, promoting the reusability of designs when developing enterprise security architectures. They provide a top-down strategy that is based on models for defining enterprise security architectures across different levels of abstraction, including their technological implementation. An enterprise security pattern combines a wide range of items. These describe generic enterprise security architectures that provide some security properties for a set of information assets in a specific context. To do this, enterprise security patterns put together in one cohesive pattern all the elements included in the enterprise security architectures: (i) the *information assets* to be protected, (ii) the *context* in which these assets are found, (iii) the *threats* associated with the assets, (iv) the *security policies*, *patterns*, *mechanisms* and *technologies* used to stop these threats, and (v) the *stakeholders* and *systems* involved in the solution. Enterprise security patterns are not intended to replace security patterns. They use the latter, integrating them into a more comprehensive pattern that can handle a greater number of threats.

By using enterprise security patterns, the Chief Information Security Officer of an organization could select a global security strategy for a specific business process, providing its designers with an optimal and proven security guideline. This would in turn standardize the design and building of the enterprise security architecture for that process. By using enterprise security patterns, security engineers could also, on one hand, manage security elements included in the different abstraction models separately. As well as this, they would be able to perform automatic transformations between them. This fact would enable the designer to carry out the selection and tailoring of security policies, patterns, mechanisms and technologies with greater ease when building enterprise security architecture for a given business process. To aid in the understanding of these patterns, we will go on to give a description of their *context model* and *solution model* in the following paragraphs.

2.1. Context model

The context of enterprise security patterns describes the generic environment under which these patterns should be applied. This context may include (i) the type of information assets to protect (*data*, *applications*, and *code and configuration*), (ii) the security realms where the assets are stored, (iii) the security policies associated with the information assets, and (iv) the general features of who (*customers*, *employees*, or *technical users*) or what (*systems*) will access them. We define here the *security realms model* and the *security policies model*.

2.1.1. Security realms model

Other work [1,18] uses the concept security domain to refer to this term. However, the term "security domain" has been adapted to have different meanings in different areas, such as physical security, and JBoss. We have thus decided to call it *security realm*, in order not to confuse the reader.

Security realms can be defined as logical and discrete entities that partition the enterprise network. The main purpose of these realms is to standardize enterprise security and thus reduce the cost, user delay, and administrative overheads of redundant security procedures. The main characteristic of security realms is that each of them has in common the same security policies. An instance of a security realm results in one or more enterprise sub-networks.

When classifying the security realms, we have taken into account the *Types of Realms* (TR) that can be found in an enterprise network, as well as who manages each of those realms, i.e., a *Trust Level* (TL). The classification of Security Realms (SR) that we propose here can be defined as *SR*: $TR \times TL$. The specific realms can be adjusted to fit different types of applications; what matters here is that we use a classification of this type. Table 1 shows with an "✓" each of the eighteen security realms provided in our classification.

Table 1
Classification of security realms.

		Trust levels		
		Managed	Externally managed	Public
Types of realms	Customer	✓	✓	✓
	Employee	✓	✓	✓
	Technical user	✓	✓	✓
	Development	✓	✓	–
	Data	✓	✓	–
	Bastion	✓	✓	–
	Transport	✓	✓	✓

When classifying the realms, we consider who manages each of them, because depending on who is responsible for security, the security policies to apply in the realm could change:

- If the realm is *Managed* (*M*), i.e., it is managed by the security department of the organization, we have the ability to design and implement security mechanisms within the realm.
- If the realm is *Externally Managed* (*EM*), i.e., it is managed by another organization or partner, we may believe that this realm has reasonable security levels, but we do not have the ability to inspect them.
- If the realm is *Public* (*P*), i.e., it is not managed by any organization, we cannot trust this realm to have appropriate security levels. We do not have the ability either to design or to implement security mechanisms within the realm.

Some of the types of security realms may be Managed, Externally Managed and Public (*Customer*, *Employee*, *Technical User* and *Transport*), while others may be only Managed or Externally Managed (*Bastion*, *Development* and *Data*), i.e., they must be managed by some organization. We provide a brief description of the types of security realms below. These types are based on the classification found in [1]:

- The *Customer* (*C*) *realm* consists of a customer, or a group of customers, who all have the same purpose. Customers typically have permission to read and modify their own data. The reading and changes that they make on the data are usually performed through specific applications. An example of this realm is a customer accessing the website from her smart phone.
- The *Employee* (*E*) *realm* consists of an employee or a group of employees who all have the same purpose. Employees tend to have permission to read and modify their data, as well as their clients' data. As customers, the reading and changes that they make on the data are usually performed through specific applications. An example of this realm is the employees of a bank working in the bank office.
- The *Technical User* (*TU*) *realm* consists of a technical user, or a group of technical users, who all have the same purpose. Technical users tend to have permission to read and modify the applications, as well as the code and configuration of the systems. The reading and changes that they make on the organization's information assets are usually performed directly on the asset. An example of this realm is the developer's team working in the building of the organization, or a developer working from an outsourced partner's building.
- The *Development* (*De*) *realm* consists of a group of applications which are under development. These applications are only accessed by technical users. Within this realm there is no data. An example of this realm is application servers used to perform the development and evaluation of new systems.
- The *Data* (*Da*) *realm* consists of a group of applications and data which are being used by customers, employees and technical users. These applications and data are in operation. An example of a data realm is mainframes.
- The *Bastion* (*B*) or *Gateway realm* consists of a group of technologies used to separate the public realms of the managed or externally managed realms. An example of this realm is the application gateway accessed by a customer.
- The *Transport* (*T*) realm consists of the parts of the enterprise network used to provide connectivity between realms. An example of a transport realm is the routers that exist between the customer's router and the first router of the organization's gateway, or the routers that exist to connect two managed data realms in different buildings belonging to the same organization.

2.1.2. Security policies model

Policies are management directives indicating a predetermined course of action, or a way to handle a problem or situation [21]. Without policies it is impossible to build secure systems; we do not know what we should protect and how much effort we should put into this protection [6]. A specific system uses a combination of security policies, according to its goals and environment. When building secure systems, designers have to consider many security policies of different types, such as confidentiality policies, integrity policies, and availability policies.

Table 2
Security policies of the Sensitivity Level (SL).

		Answers combinations			
SL	Security policies	1	2	3	4
4	Secure Channel (SC) & Hidden Storage (HS)	Yes	Yes	Yes	Yes
3	Secure Channel (SC) & Clear Storage (CS)	Yes	Yes	Yes	No
5	Secure Channel (SC) & Blocked Storage (BS)	Yes	Yes	No	-
-	Clear Channel (CC) & Hidden Storage (HS)	Yes	No	Yes	Yes
1	Clear Channel (CC) & Clear Storage (CS)	Yes	No	Yes	No
2	Clear Channel (CC) & Blocked Storage (BS)	Yes	No	No	-
6	Blocked Channel (BC)	No	-	-	-

The security level of the policies applied in each realm can vary, but we need to preserve all the required security attributes of assets (confidentiality, integrity, availability and auditability) when they are handled or transferred in a realm. We have defined a group of security policies associated with the confidentiality that the enterprise security patterns will use to define the sensitivity level of an information asset. To achieve this, we have combined the answer to four dependent questions related to the following security aspects: *access authorization*, *encryption*, and *storage authorization*. The four questions are listed below:

1. Can the information asset A be transported through the security realm Sr?
2. If so, should A be encrypted?
3. Can A be stored in Sr?
4. If so, should A be stored in a concealed form?

Table 2 shows the possible answers, the security policies associated with each combination, and a number that denotes the Sensitivity Level (SL) provided by each policy (1 is the lowest and 6 the highest). This number will help us when designing the solutions of the enterprise security patterns.

After removing the policy *Clear Channel* & *Hidden Storage* (*CC* & *HS*, shaded row), we obtain six different security policies. The policy *CC* & *HS* has been removed, because it is not usual to find a security policy in which the information assets require encrypted storage while being transported in an open form. When protecting the information assets, enterprise security patterns will use additional security policies, such as integrity policies, availability policies, and auditability policies.

Several distinct organizations might apply different sensitivity levels to the same asset. For example, when classifying the customers' account, a food industry organization could decide to apply security policies with low or medium security level in all its security realms. However, a banking organization may decide to apply security policies with a high or very high security level. Because of such possibilities, enterprise security patterns do not try to protect single information assets. Their goal is to protect information assets with the same sensitivity level in a particular context.

2.2. Solution model

The solution of an enterprise security pattern is a model-driven architecture instance, which provides a solution to recurring problems

related to information systems security. The four complementary models or viewpoints included in this solution are: the *Computationally Independent Model* (CIM), the *Platform Independent Model* (PIM), the *Platform Specific Model* (PSM), and the *Product Dependent Model* (PDM). We discuss each of these below.

Computationally Independent Model: this model provides a description of the security policies that the system should enforce, independently of that system's functional and technological characteristics. The security policies should be applied to the information assets and security realms. When building secure systems, the CIM could help us to define the security requirements of the systems to be protected.

Platform Independent Model: this model provides a conceptual description of the security mechanisms that should be incorporated into the system, along with the relationships that exist among them, independently of the system's technological characteristics and implementation detail. The same CIM could be instantiated N times in this model, since a security policy may correspond to different security patterns. Good guidelines which can be used as a basis to select the security patterns needed are those developed by Schumacher et al. in [17] or Fernandez in [5]. When building secure systems, the PIM could help us to use the enterprise security patterns in the analysis stages of the security methodologies.

Platform Specific Model: this model defines the architectural components included in the enterprise security architecture, independently of the technology used to solve the problem. The PSM should take into account how to place the security mechanisms within the architecture. The same PIM can be instantiated N times in this model, since a security mechanism may be placed in different architectural components. The security patterns described in the PIM are included within architectural security components. Two good guidelines which can be used as a basis for selecting the architectural component are the ISO/IEC-27000-series [10] and the IT Baseline Protection Manual [2]. The PSM could help us to use the enterprise security patterns in the design stages of the security methodologies.

Product Dependent Model: The PSM needs to be installed in a specific technological architecture. The same PSM could be instantiated N times, since the same architectural component may correspond to different technological products. The technological products must be reputable ones, made by well-known manufacturers in the security industry. The final solution may vary significantly, depending on the technologies used.

Using this solution approach, we could, on the one hand, manage security elements included in the different abstraction models separately, and on the other hand, perform automatic transformations between them. For example, if we have the CIM of the solution, we may manage the security policies needed to assure the information assets, without considering the security patterns. In addition, we may transform the security policies included in the CIM automatically, thereby obtaining the security patterns used in the PIM of the solution.

3. An enterprise security pattern: secure software as a service

We document here an enterprise security pattern which could be used by organizations to protect the information assets when using outsourced online applications: for example, *Google Apps for business*. We will go on to discuss each of the sections included in the pattern template. This template includes sections of the one provided by Buschmann et al. [3], along with some new sections that we consider necessary when designing enterprise security architectures, such as *Intent*, *Known Incidents* and *Considerations*.

3.1. Intent

This pattern attempts to protect the confidentiality of the data included in the outsourced online applications of an organization.

3.2. Context

Employees of an organization access from home (*Public Employee realm*, P-E) outsourced online applications. The service provider has placed the applications in a data center (*Externally Managed Data realm*, EM-D). Employees access the service provider through Internet (*Public Transport realm*, P-T). The service provider has an application gateway (*Externally Managed Bastion realm*, EM-B) between the Internet and its data center.

To allow employees to continue using the access credentials that they employ in their organization, when an employee tries to access the online applications, the service provider redirects the employee's browser to the organization's gateway (*Managed Bastion realm*, M-B). At that moment, employees validate their credentials in the organization's systems (*Managed Data realm*, M-Da) to get a ticket to access the provider. Once the employee has the ticket s/he can access his/her online applications. Fig. 1 shows the context diagram of this pattern.

The Sensitivity Level (see Section 2.1.2) of the information assets (data) that this pattern attempts to protect is shown in Table 3. As we can observe in Table 3, the data included in the outsourced applications should only be stored in a clear form by the employee (P-E). The organization's data center (M-Da) could store the data in a clear form, but these data are not related to the outsourced applications. The rest of the realms should store the data in a hidden way. The data may leave the service provider, but the communication channels should be secure. The organization's data center (M-Da) may transport data in a clear form, but these data are not related to the outsourced applications.

This pattern should only be used when the organization's employees use the outsourced online applications to store information assets that meet this sensitivity level. Organizations should ensure that no employee stores information assets with higher sensitivity levels.

3.3. Problem

In the past, mailbox and collaboration applications used by the organizations were placed within the organization. Employees accessed them from their home through the Internet. This context provoked a set of threats that the organizations had to handle. Some threats (those related to integrity and availability should also be handled) related to the confidentiality of data in this environment are:

- An attacker may read the data accessed by the employee via Internet. To prevent this, the organizations need to ensure that the communications between the employee and the data center are secure.
- An attacker may steal an employee's identity and access his/her applications. To prevent this, organizations need to ensure that the employee is who s/he claims to be.
- An attacker may take advantage of a vulnerability to access an employee's applications. To prevent this, organizations constantly need to patch the applications and the servers where the applications are hosted.
- A technical user who performs application maintenance may leak information. To prevent this, organizations need to ensure that the data are stored in a hidden form.

Since the birth of the cloud computing paradigm, the context for accessing organizations' applications has changed significantly. Furthermore, some companies have taken on the provision and maintenance of online applications of other organizations as their main objective. The threats that the organizations have to handle in these contexts are similar to those seen in the context in which the employees accessed the organization's systems (listed previously); the way of handling them is different, however.

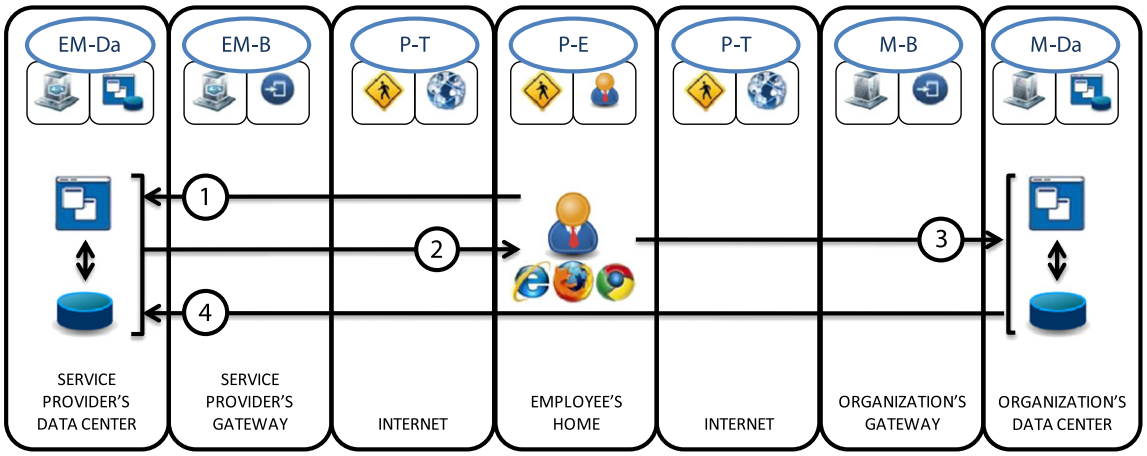


Fig. 1. Context diagram.

3.4. Known incidents

There are many incidents of identity thefts every day. The main objective of these thefts is to obtain relevant information from the person (or company) attacked, or to steal his/her (or its) identity. One of the most notorious incidents was the case of when hackers stole the access credentials of the Twitter account owned by Fox News; they then published the news that President Barack Obama was dead [8].

All companies are exposed to this type of theft. By using the pattern that we are describing here, companies could prevent hackers from accessing the employees' online applications, even if they manage to steal their identity. This is because the solution provided by the pattern ensures that the employee really is who s/he claims to be (this is explained in more detail in the next section).

3.5. Solution

In the paragraphs below we discuss each of the models included in the solution:

Computationally Independent Model: we need to apply here the security policies included in the sensitivity level of the information assets. As shown in the diagram of the CIM (Fig. 2), we could prevent an attacker from intercepting the application data, encrypting the channels and storing the data in a concealed way.

Platform Independent Model: here we realize the security policies of the CIM as security patterns. All security patterns included in the PIM are described in [17], except *Security Logger/Auditor* as described in [7] and *Hidden Storage*, which has not been described yet. The security pattern *Hidden Storage* should ensure that nobody who is unauthorized can read the information stored. Instantiations of the same pattern P are denoted as P_1, P_2, etc. The types of channels that we may find within the PIM are: *clear channel* (single line), and *secure channel* (double line). In addition, these channels show a logical representation of the type of message that they transport. The type of messages that could be transported are: *request or response message* (solid line), and *record message* (dashed line).

Table 3
Sensitivity Level (SL) of the information assets.

Security realms	Security policies	SL
Externally Managed Data	*Secure Channel* (SC) & *Hidden Storage* (HS)	4
Externally Managed Bastion	*Secure Channel* (SC) & *Hidden Storage* (HS)	4
Public Transport	*Secure Channel* (SC) & *Hidden Storage* (HS)	4
Public Employee	*Secure Channel* (SC) & *Clear Storage* (HS)	3
Managed Bastion	*Secure Channel* (SC) & *Hidden Storage* (BS)	4
Managed Data	*Clear Channel* (CC) & *Clear Storage* (CS)	1

We set out the sequence of actions shown in the PIM diagram (Fig. 3) in the following points:

1. The employee requests a secure channel through a browser to access his/her online applications (*Secure Channel_1*).
2. The service provider checks the employee's organization.
3. The service provider redirects the employee's browser to the organization's systems, including an access ticket associated with the employee.
4. The employee's browser requests a secure channel to access the organization's systems (*Secure Channel_2*).
5. The employee provides his/her credentials in the organization's systems (*Identification_2*).
6. The organization checks that the employee is who s/he claims to be (*Authentication*). If the validation is successful, the employee obtains the signed ticket to access his/her online applications.
7. The organization's systems redirect the employee's browser to the service provider's systems, along with the signed access ticket. *Access Control* checks if the signed ticket is the same as the one generated previously.
8. If the access ticket is valid, the employee is able to access his/her applications.
9. The applications must decode the data so that these can be shown to the employee (*Hidden Storage*).
10. Before showing the online applications to the employee, the service provider rechecks that the employee has permission to access those data and applications (*Access Control*).
11. The service provider shows the employee his/her online applications through a *Secure Channel*.
12. The service provider stores a session cookie in the employee's browser. From this point on, the employee may access his/her online applications without re-authentication.

In order to audit possible attacks, the mechanisms for identification, authentication, access control, and hidden storage should record all activity in the security patterns *Security Logger/Auditor*.

Platform Specific Model: here we transform the security patterns of the PIM into architectural components. As shown in the diagram of the PSM (Fig. 4), the security pattern instantiations *Secure Channel_1* and *Identification_1* are transformed into a *Web Server*. The security pattern instantiations *Secure Channel_2* and *Identification_2* are transformed into a *Reverse Proxy*. The security pattern instantiation *Security Logger/Auditor* is changed into a *Log System*. The security pattern instantiation *Access Control* and the applications are transformed into an *Application Server*. Finally, the security pattern instantiation *Hidden Storage* and the applications' data are changed into a *Dissociation Data Server*.

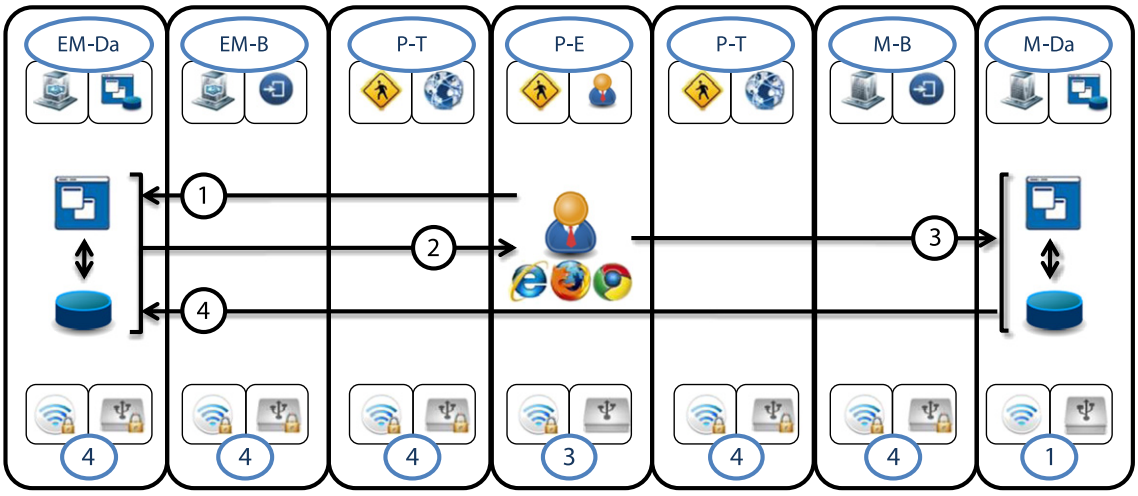


Fig. 2. Computationally independent model (CIM) diagram.

In order to prevent an attacker from accessing an employee's applications after stealing his/her identity; we need to ensure that the employee is who s/he claims to be. To do this, an authentication system with a high security level should be used (something stronger than passwords). In the pattern solution, we have decided to include a *Token-Based Authentication Server*, because it is currently more widespread in use, but we could also have used biometric proof, or some other kind of strong authentication.

Product Dependent Model: here we transform the architectural security components into technological products. The diagram of the PDM (Fig. 5) shows the technological products that we have decided to include in the solution. We have selected these technologies because we consider them reputable and they are currently used by many organizations; we could have selected another different set of technologies, however.

3.6. Considerations

The considerations of the pattern take into account the technologies selected in the PDM of the solution. Another different set of technologies could change this analysis. Table 4 shows the result of the analysis for each of the relevant aspects.

We can see in Table 4 that, when deploying the solution, the performance overheads of the enterprise security architecture do not increase; it could even decrease because part of the IT infrastructure would be outsourced. In some cases, the security and log administrator has to work outside the organization. This may mean a small increase in the size of the personnel in the security and log team. The installation cost does not increase; it might even decrease, because IT infrastructure, IT personnel, implementation, and maintenance are outsourced. Once the solution is deployed, the residual risk is minimal. This means that the solution does not need complementary measures to attain its initial objective.

3.7. Consequences

As we said previously, threats found in the problem are related to the confidentiality of the assets. Integrity and availability threats could be handled in similar ways. The following points outline the security mechanisms that we have included in the pattern to prevent or reduce the risk of the threats identified:

- An attacker may read the data accessed by the employee via Internet. The service provider's *Web Server* and the organization's *Reverse Proxy* can prevent this by using secure channels.
- An attacker may access an employee's applications, after stealing his/her identity. To prevent this, we include a *Token-based Authentication Server*.
- An attacker may use a vulnerability to access an employee's applications. To prevent this, the service provider constantly has to patch

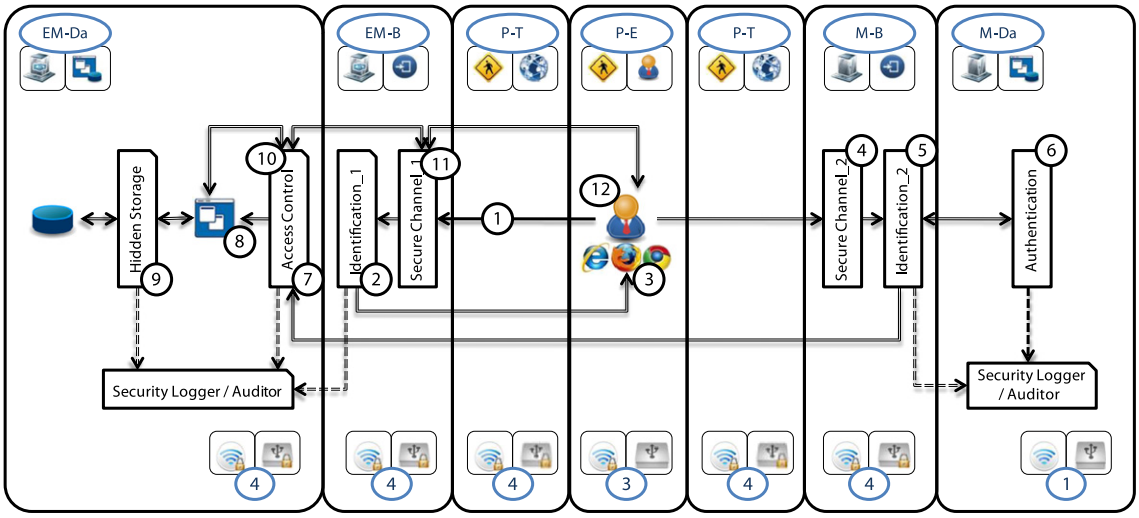


Fig. 3. Platform independent model (PIM) diagram.

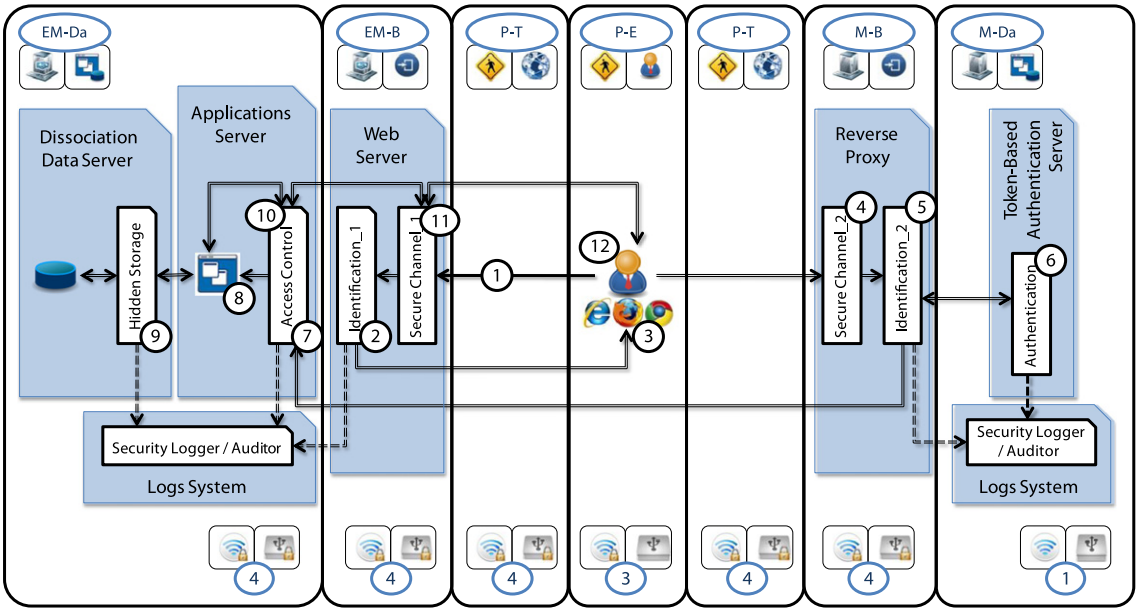


Fig. 4. Platform specific model (PSM) diagram.

the applications and the servers where the applications are hosted.

- A technical user who performs the applications maintenance may leak information. To prevent this, we include a *Dissociation Data Server* in the service provider.

This pattern would also be applicable in a context where the employees access their online applications from the organization's premises (Managed Employee realm, M-E), rather than from their home.

3.8. Known uses

As shown previously, Google is a provider of online applications that offer the security measures and architecture included in the pattern solution. One of their most popular products is *Google Apps*. Forty million active users and four million businesses are currently using it [9]; these include Florida Atlantic University and the BBVA Group.

4. Enterprise security pattern: a model-driven architecture instance

Model-Driven Architecture (MDA) defines an approach to IT system specification. It separates the specification of system functionality from the specification of the implementation of that functionality on a particular technology platform [12]. MDA defines an architecture for models, providing a set of guidelines for structuring specifications expressed as models.

It has already been commented that the solution of an enterprise security pattern is an MDA instance. As shown in Fig. 6, each of the solution's models (CIM, PIM, PSM, and PDM) can be instantiated N times into the next model in the hierarchy. This means that the same CIM may have diverse PIM associated, the same PIM may have different PSM associated with it, and the same PSM may have a variety of PDM. In addition, each of the resulting PDM may have different security levels, i.e., the risks assumed by the organization when implementing the pattern may be different.

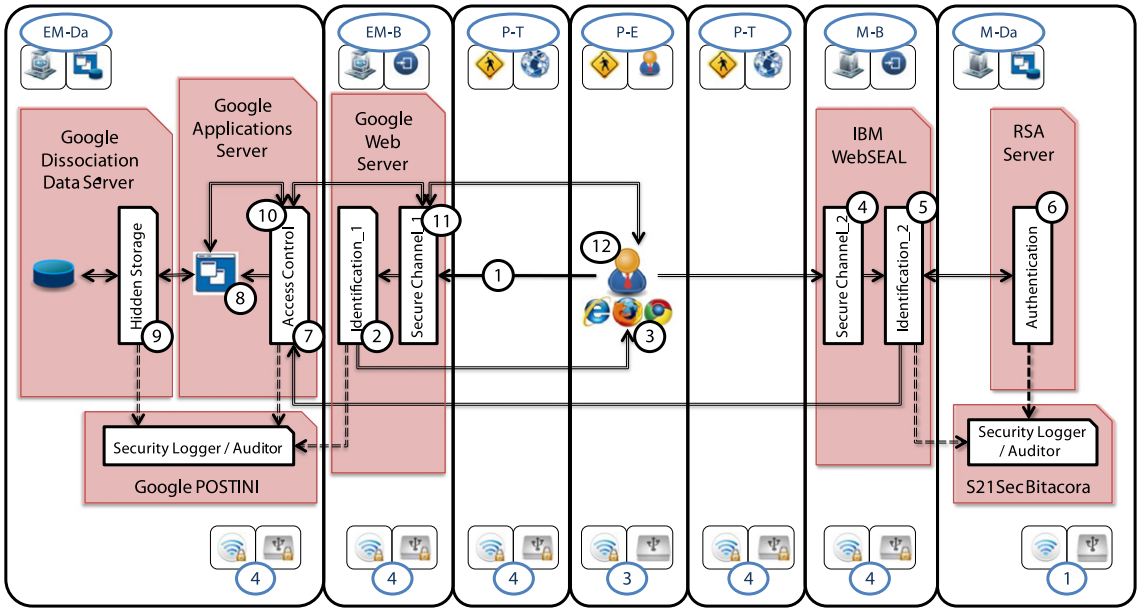


Fig. 5. Product Dependent Model (PDM) diagram.

Table 4
Considerations.

	Aspects to consider	Analysis
Performance overhead	Storage	0
	Primary memory	0
	Processor	0
	Bandwidth	0
Complexity	Security administrator	1
	Log administrator	1
	End user	0
	Massive expansion	0
	System administrator	0
	Installation cost	0
	Residual risk	0

When defining the *Enterprise Security Pattern*: *Secure Software as a Service* (see Section 3), we have tried to adopt the particular transformations between models that are needed to obtain an enterprise security architecture that reduces or mitigates the largest number of risks. In this section, we present different instances of the solution models of the enterprise security pattern *Secure Software as a Service*, in order to check what risks an organization would assume if each of the instances was deployed. To do this (see Fig. 6), the *CIM* has been transformed into *two PIM instances*; the *PIM Instance 1* has been transformed into *three PSM instances*; finally, the *PSM Instance 1* has been transformed into *three PDM instances*.

With this approach, we intend to show how the design decisions adopted when performing the transformations between the solution's models can have a direct impact on the security provided by the pattern.

4.1. Second instance for the platform independent model (PIM)

From the CIM diagram presented in Section 3 (Fig. 2), we have defined a second instance for the PIM (Fig. 7). This new diagram is similar to the PIM shown in Section 3 (Fig. 3), except that we have removed the security pattern *Security Logger/Auditor*, as well as both the service provider's data center and the organization's data center.

This small change would significantly reduce the security level of the pattern's solution, because the organization could not investigate or audit possible attacks; even worse, they could not find out if the enterprise security architecture deployed is working properly.

4.2. Second instance for the platform specific model (PSM)

From the PIM diagram presented in Section 3 (Fig. 3), we have defined a second instance for the PSM (Fig. 8). This new diagram is similar to the PSM shown in Section 3 (Fig. 4), except that the authentication server located in the organization's data center is based on *User and Password* (U&P), instead of *Token*.

This small change would also reduce the security level of the pattern solution, because the organization could not ensure that an employee actually is who s/he claims to be. In this particular case, an attacker could impersonate an employee if he/she obtained the access credential (for example, using a phishing attack).

4.3. Third instance for the platform specific model (PSM)

From the PIM diagram presented in Section 3 (Fig. 3), we have defined another PSM diagram (Fig. 9). This new diagram is similar to the

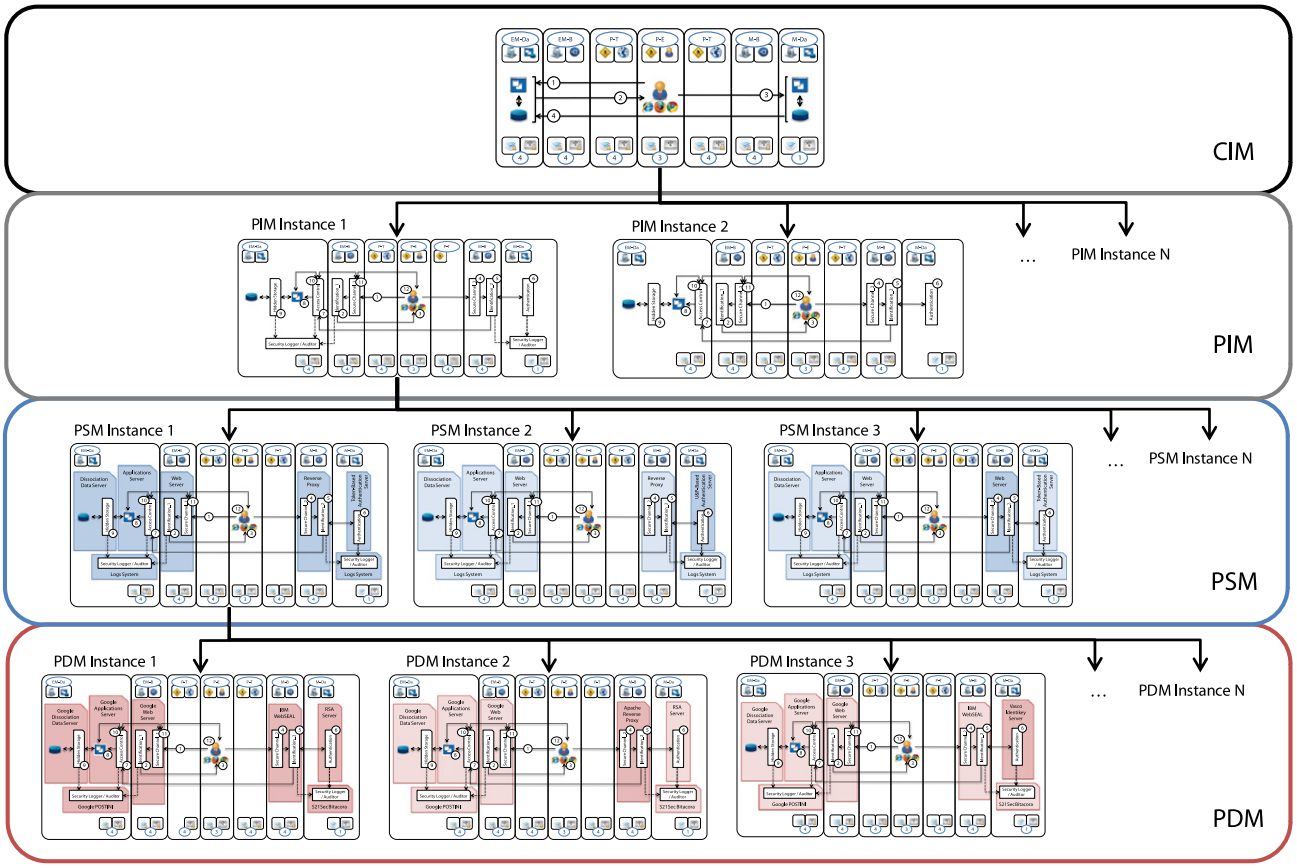


Fig. 6. A model-driven architecture for the enterprise security pattern: secure SaaS.

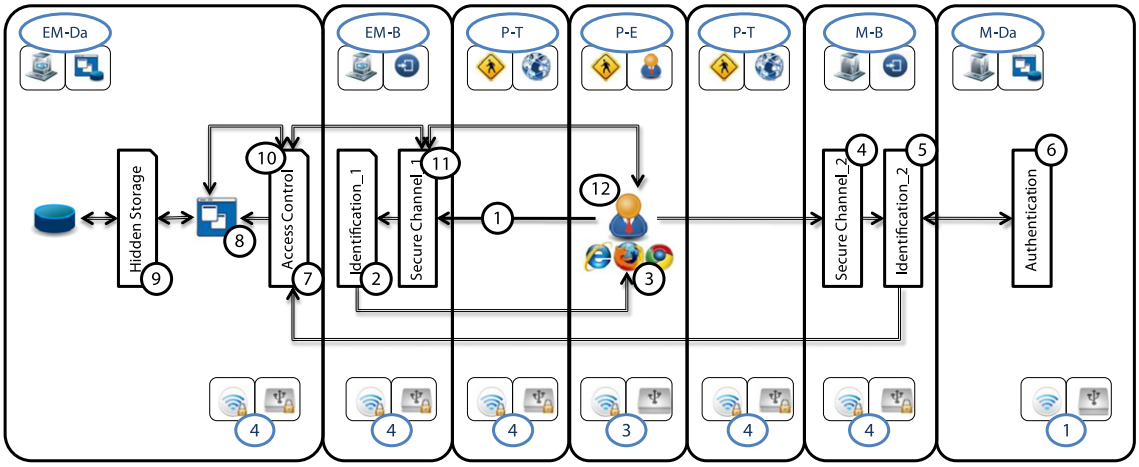


Fig. 7. Second instance for the platform independent model (PIM) diagram.

PSM shown in Section 3 (Fig. 4), except that the *Reverse Proxy* located in the organization's data center has been replaced by a *Web Server*.

This small change would also reduce the security level of the pattern solution, because the *Web Server* usually has information assets that can be accessed from outside the organization. If an attacker finds vulnerability in the Web Server, the information stored could be exposed.

However, the *Reverse Proxy* does not store information assets, because it only works as an access gateway. If an attacker finds vulnerability in the *Reverse Proxy*, s/he could not leak information.

4.4. Second instance for the Product Dependent Model (PDM)

From the PSM diagram presented in Section 3 (Fig. 4), we have defined another PDM diagram (Fig. 10). This new diagram is similar to the PDM shown in Section 3 (Fig. 5), except that the *IBM WebSEAL* located in the organization's data center has been replaced by an *Apache Reverse Proxy*.

In contrast to the other cases, this small change would not significantly reduce the security level of the pattern solution. The main difference between security technologies is usually related to the precision and performance that they offer. In this particular case, when retrieving resources on behalf of an authenticated user from one or more servers, *Apache Reverse Proxy* could only filter URLs, while *IBM WebSEAL* could filter the variables included in the URLs.

4.5. Third instance for the Product Dependent Model (PDM)

From the PSM diagram presented in Section 3 (Fig. 4), we have defined another PDM diagram (Fig. 11). This new diagram is similar to the PDM shown in Section 3 (Fig. 5), except that the *RSA Server* located in the organization's data center has been replaced by a *VASCO IDENTIKEY Server*.

In this particular case, this small change would not significantly reduce the security level of the pattern solution. The use of one particular server or another does not depend on the security level, because both of the manufacturers use validated mechanisms that are reliable.

5. Conclusions and future work

Security patterns are a good way to construct and evaluate new security mechanisms, but they are not applied as much as they could be, because designers have problems in selecting them and applying them in the right places. Enterprise security patterns could improve the application of the patterns by incorporating them in a more

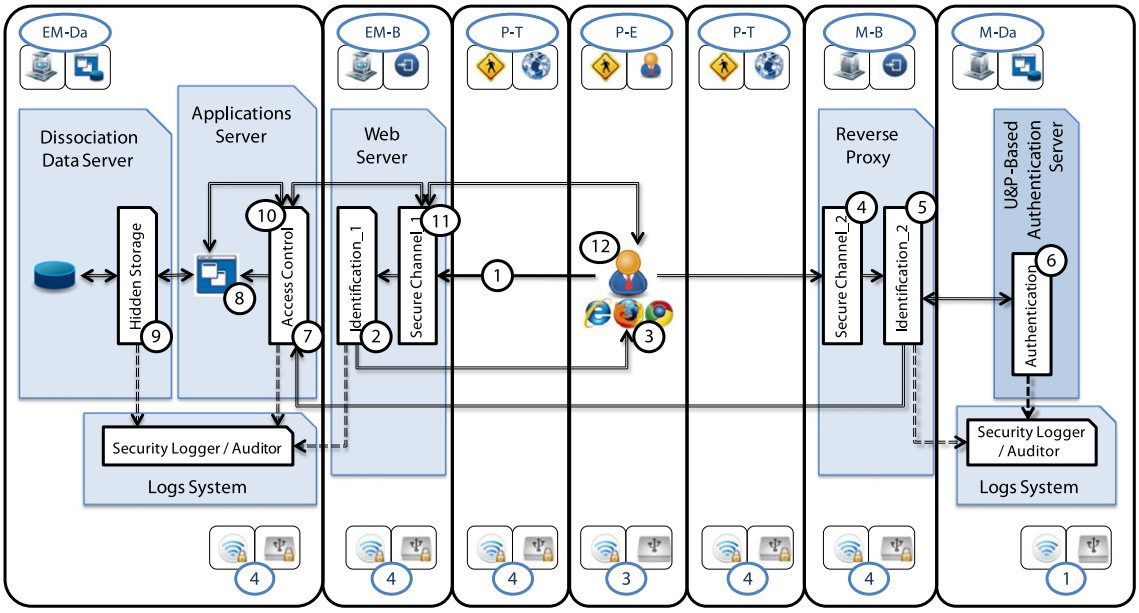


Fig. 8. Second instance for the platform specific model (PSM) diagram.

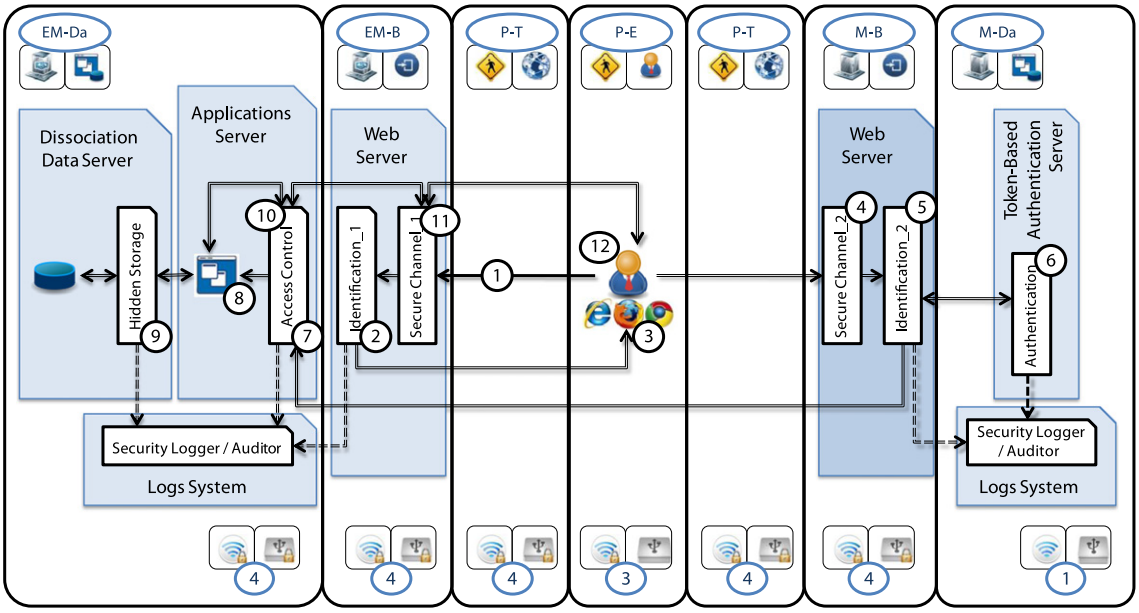


Fig. 9. Third instance for the platform specific model (PSM) diagram.

comprehensive pattern that may handle more threats. There would thus be a smaller number of enterprise security patterns, making it simpler for designers to choose which to use.

The solution of an enterprise security pattern offers a model-driven architecture instance. This means that when defining a new pattern, designers should perform the transformations between the solution models while at the same time trying to remove or mitigate as high a number of risks as possible. A small change in one of the solution models may significantly reduce the security level of the pattern.

Using Software as a Service (SaaS) has become very popular in recent times. This wide acceptance has come about from the fact that companies may save money on support and upgrade costs, IT infrastructure, IT personnel, implementation, and maintenance. However, before using SaaS, organizations need to be aware of a variety of threats and be prepared to handle them.

When adopting SaaS, organizations could consult the enterprise security pattern that we have presented here, in an effort to protect the data included in the outsourced applications from a common set of threats. Organizations which have already adopted SaaS could also consult this pattern to find out if they are protecting their information assets correctly. As future work, we intend to document further enterprise security patterns which can be consulted by organizations when using Platform as a Service (PaaS) or Infrastructure as a Service (IaaS).

Acknowledgments

This research has been hosted at Florida Atlantic University (FAU) and carried out in the framework of the following projects: MASAI (TIN2011-22618), financed by the Spanish Ministry of Education and Science, and FEDER, SIGMA-CC (TIN2012-36904) and GEODAS (TIN2012-37493-C03-01), financed by the "Ministerio de Economía y Competitividad" (Spain).

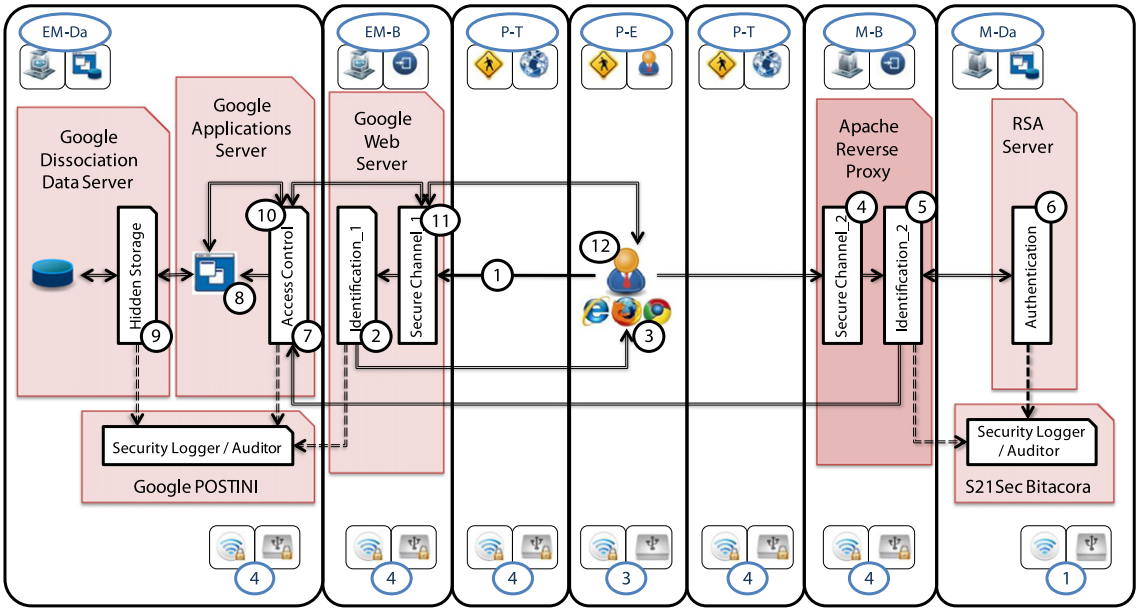


Fig. 10. Second instance for the Product Dependent Model (PDM) diagram.

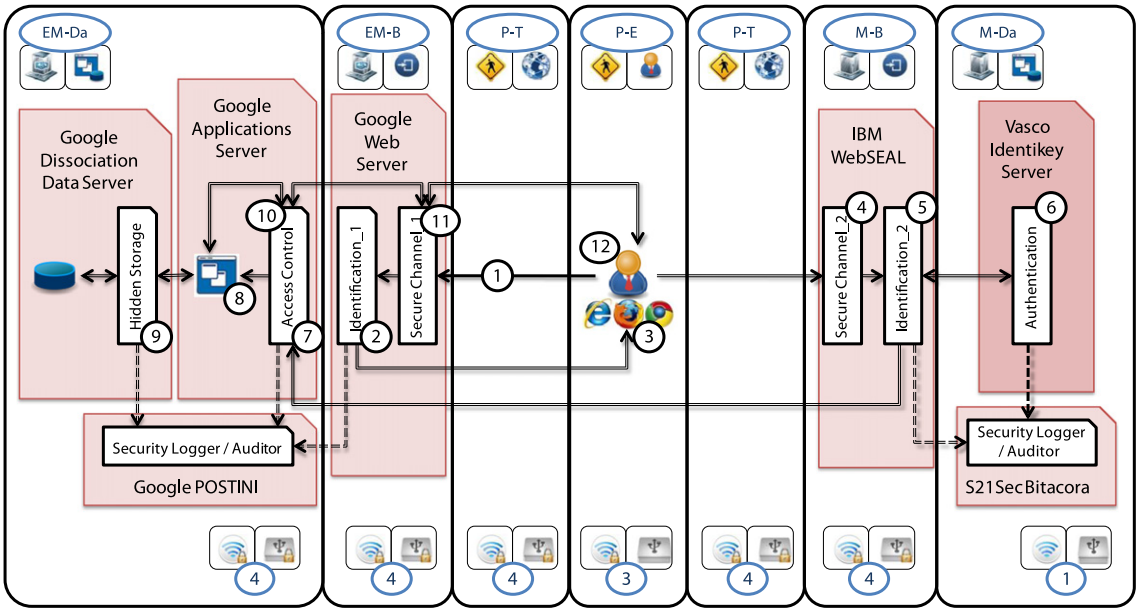
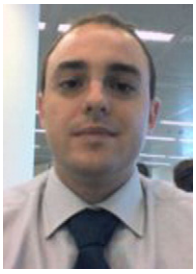


Fig. 11. Third instance for the Product Dependent Model (PDM) diagram.

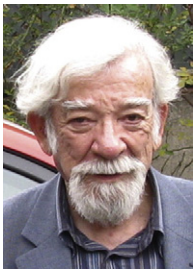
References

[1] N. Arconati, One Approach to Enterprise Security Architecture, SANS Institute, 2002.
[2] BSI, IT Baseline Protection Manual, Federal Agency for Security in Information Technology, Germany, 2000.
[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-oriented Software Architecture: a System of Patterns, Wiley, 1996.
[4] J. Espadas, D. Concha, A. Molina, Application development over software-as-a-service platforms, The Third International Conference on Software Engineering Advances, 2008.
[5] Fernandez, E.B., *Security patterns in practice*: *building secure architectures using software patterns*. To appear in the Wiley Series on Software Design Patterns, Under contract with J. Wiley.
[6] Fernandez, E.B.; Gudes, E.; Olivier, M., Policies and Models. In: *The design of secure systems*. Under contract with Addison-Wesley.
[7] E.B. Fernandez, S. Mujica, F. Valenzuela, Two security patterns: least privilege and security logger/auditor, Asian PLoP, 2011.
[8] GIZMODO, Fox News' Twitter Account Hacked, Retrieved: February, 2013; Available from: http://gizmodo.com/5817870/fox-news-twitter-account-hacked-claims-barack-obama-is-dead.
[9] Google, Businesses Share their Stories — Google Apps, Retrieved: February, 2013; Available from: http://www.google.com/apps/intl/en/customers/index.html.
[10] ISO, International Organization for Standardization, Retrieved: February, 2013; Available from: http://www.iso.org.
[11] D. Ma, The business model of software-as-a-service, IEEE International Conference on Services Computing (SCC 2007), 2007.
[12] J. Miller, J. Mukerji, MDA Guide Version 1.0, 2003.
[13] S. Moral-García, S. Moral-Rubio, E.B. Fernandez, E. Fernández-Medina, Enterprise security pattern: a new type of security pattern, Submitted to the Security and Communication Networks Journal, Wiley, 2012.
[14] S. Moral-García, S. Moral-Rubio, E.B. Fernandez, E. Fernández-Medina, A new enterprise security pattern: secure software as a service (SaaS), The 9th International Workshop on Security in Information Systems (WOSIS) co-located with ICEIS, 2012, Wroclaw (Poland), 2012, pp. 14–26.
[15] R. Ortiz, S. Moral-García, S. Moral-Rubio, B. Vela, J. Garzás, E. Fernández-Medina, Applicability of security patterns, International Symposium on Information Security (OnTheMove Conferences), Springer, Crete (Greece), 2010, pp. 672–684.
[16] I. Ray, R.B. France, N. Li, G. Georg, An aspect-based approach to modeling access control concerns, J. Inf. Softw. Technol. 46 (9) (July 2004) 575–587.
[17] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, P. Sommerlad, Security Patterns: Integrating Security and Systems Engineering, Wiley, 2006.
[18] J. Sherwood, A. Clark, D. Lynas, Enterprise security architecture, SABSA White Paper2009.
[19] M. Turner, D. Budgen, P. Brereton, Turning software into a service, Computer 36 (10) (2003) 38–44.
[20] Uzunov, A.V.; Fernandez, E.B.; Falkner K., *Engineering security into distributed systems*: *a survey of methodologies*. Accepted for the Journal of Universal Computer Science.
[21] C.C. Wood, Information Security Policies Made Easy, 2000. (Version 7).



Santiago Moral-García is a professor in the Department of Computer Languages and Systems II at Rey Juan Carlos University in Madrid, Spain. He holds a MS degree in Decision Engineering from the same Rey Juan Carlos University. Moral-García is currently preparing his PhD on the construction of enterprise security architectures based on model-driven development and automatic transformations. He has published various papers in international conferences and workshops (OTM, PATTERNS, WOSIS, etc.). He is the author of a manuscript journal (Advances in Security).

Santiago Moral-Rubio is an IT Engineer by the Universidad Politécnica de Madrid, in Spain. He holds two MS degrees, one in Information Systems and another in Decision Engineering from Rey Juan Carlos University in Madrid, Spain. Moral-Rubio is ISACA certified as CISA, CISM and CGEIT. He is currently preparing the Doctorate in Risk Management of Intentionality. In the middle 80s, he founded his own company named Open Systems Administration Group. Since 2001, Moral-Rubio serves as BBVA Group CISO. He is a frequent speaker in Spanish and Latin-American's Information Security and Fraud Prevention for the Financial Industry Forums.



Eduardo B. Fernandez (Eduardo Fernandez-Buglioni) is a professor at Florida Atlantic University in Boca Raton, Florida. He has published numerous papers on authorization models, object-oriented analysis and design, and security patterns. He has written four books on these subjects, the most recent being a book on security patterns. He has lectured all over the world at both academic and industrial meetings. He has created and taught several graduate and undergraduate courses and industrial tutorials. He holds a MS degree in Electrical Engineering from Purdue University and a Ph.D. in Computer Science from UCLA.

Eduardo Fernández-Medina holds a PhD and a MS degree in Computer Science from the University of Sevilla. He is a co-editor of several books and chapter books on security in information systems, and has papers in international conferences (BPM, UML, ER, ESORICS, TRUSTBUS, etc.). He is the author of several manuscript journals (Decision Support Systems, Information Systems, ACM Sigmod Record, Information Software Technology, Computers & Security, Computer Standards and Interfaces, etc.). He leads the GSyA research group at the University of Castilla-La Mancha, in Ciudad Real, Spain. He belongs to various professional and research associations (ATI, AEC, AENOR, IFIP WG11.3, etc.).