TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 774 – 792
© TÜBİTAK
doi:10.3906/elk-1303-12

An improved security framework for Web service-based resources

Wenbin JIANG*, **Hui XU**, **Hao DONG**, **Hai JIN**, **Xiaofei LIAO**
Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, P.R. China

Received: 02.03.2013 • **Accepted/Published Online:** 30.09.2013 • **Final Version:** 23.03.2016

Abstract: Web service-based application has become one of the dominative ones of the Internet. This trend brings more and more security challenges in reliability, confidentiality, and data nonrepudiation, especially in some systems that have massive diversified resources. An improved framework for secure accesses of Web resources is presented and implemented by extending and enhancing the Spring Security framework. It improves the security level of systems for identity authentication, authorized access, and secure transmission. The highly safe authentication is based on the integration of an improved authentication module of Spring Security with a U-key method and a RSA algorithm. For authorized access, the Spring Security's ACL (access control list) mechanism is improved by optimizing the domain object-level access control. For secure transmission, a compromising method is presented to take both the security level and the speed of data transmission into account by means of mixing the RSA and DES algorithms. In addition, the security interceptor of Spring Security is extended and a series of security filters are added to keep Web attacks away. The above improved security framework has been applied to an online virtual experiment platform named VeePalms. The experimental results show that most security problems with high severity in the system have been solved and medium-low severe problems decreased dramatically. Moreover, VeePalms has been used in practice for about 2 years, which has proved the effectiveness of the security framework.

Key words: Web service, Spring Security, authentication, authorized access, secure transmission

1. Introduction

More and more network applications in various fields such as e-learning, electronic business, and e-governance tend to apply Web services to build systems and provide Web resources for users. Although availability is one of the essential issues of Web services, some other key issues, such as security, reliability, and interoperability, are also indispensable for enterprise applications based on Web services. Among all these issues, security is an inevitable key issue for all applications. Only secure Web services can guarantee the safety of diversified Web applications. However, at present, SQL (Structured Query Language) injection, cross-site scripting, and other Web service attacks are still constantly emerging because of various security holes [1]. Most communications between applications based on Web services require the trust of their partners, which reduces the wide spread of Web services [2]. Since Web services are based on XML (Extensible Markup Language) and rely on the Internet for information exchange, there are considerable numbers of unsolved security problems, such as data eavesdropping, wiretapping, and illegal access, which bring about serious potential safety risks in data exchange and transmission [3].

*Correspondence: wenbinjiang@hust.edu.cn

VeePalms is a representative Web service-based system developed by our lab [4–6], which provides multidiscipline virtual experiments for massive learners on a unified platform. It can support tens of thousands of students from all over China to do experiments concurrently with the aid of Web services and high-performance distributed computing technologies.

As a system for serving tens of thousands of users concurrently, VeePalms needs to manage and store a lot of various structured and unstructured data, such as information of users, XML (Extensible Markup Language) experiment components, scenes, and experiment reports [6]. It is a challenge to design and implement a practical and efficient Web security framework for this type of system.

In this paper, an improved security framework is presented by enhancing and extending the Spring Security framework [7,8]. It can ensure the security requirements of Web services by combining identity authentication, authorized access, and secure transmission. At the same time, some security interceptors of the security architecture are modified so that some kinds of Web attacks are prevented from damaging the system. It has been applied in VeePalms to prove its effectiveness. This security framework also has good independence and compatibility since it derives from Spring Security. It can be used for other Web service-based systems.

This paper is organized as follows: Section 2 presents some related work. The design and implementation of the presented security framework are discussed in Section 3 in detail. Section 4 provides performance evaluation and a discussion. Finally, Section 5 concludes the work.

2. Related work

2.1. Web service security

So far, security researches of Web services mainly focus on establishment of service safety regulations, combined with Web service protocol stacks. For example, many researchers built new safety mechanisms by modifying and extending SOAP (Simple Object Access Protocol) [9]. A number of standard organizations, companies, and social groups have carried out a lot of research work. Some organizations such as W3C (World Wide Web Consortium), IETF (Internet Engineering Task Force), and OASIS (Organization for the Advancement of Structured Information Standards) set a series of standards for XML and Web service safety [10]. W3C issued the XML encryption standard in 2002. The standard achieved encryption for a selected part of specific data, but it has not been used as a unified and standardized specification widely. The OASIS organization released the WS-Security (Web Service Security) standard in 2004. WS-Security is a network transmission protocol that provides methods for application security based on Web services.

Most of deployed Web services rely on the security mechanism in the transport layer by using the combined authentication of SSL (Secure Sockets Layer) and HTTP (Hypertext Transfer Protocol), which can provide security guarantees for Web service to some extent. SSL, TLS (Transport Layer Security), and IPSec (Internet Protocol Security) [11] are often used for email, e-commerce, bank websites, etc. Some international mail services, such as Gmail, Yahoo Mail, and Live, adopt SSL security mechanisms (without certified authority from a third party). These large Web applications generally apply direct point-to-point data transmission without the participation of third party services, which can ensure the nonrepudiation of these applications. However, this simple and one-fold security solution is short of the following characteristics: end-to-end protection, selective protection, sound and flexible authentication mechanism, and environmental support of message-level safety. For example, SSL/TLS can only encrypt all SOAP messages, instead of the selected ones. One thing a comprehensive Web service security system needs is an end-to-end security mechanism [12].

2.2. Access control

The goal of access control is to restrict the rights and scope of user access to resources through certain methods. Generally, according to differences of access control strategies, access control is divided into three types: discretionary access control (DAC), mandatory access control (MAC), and role-based access control (RBAC) [13,14].

RBAC done is to appoint some access rights to a certain role first. Then a user can win all access permissions of the role. It has been widely used.

Access control of Web service resources can be generally divided into the following three types: the access control of URL (Uniform Resource Locator) resources, that of operation methods, and that of domain objects. For access control of domain objects, it can be achieved in the database by writing a set of triggers for each data sheet, which can achieve row-level data access. However, because many tables are required to be handled while changing permissions, a number of codes involved should be altered, which also reduces the maintainability of security system.

Our research focuses on the improvement of the RBAC, and especially the access control of domain objects.

2.3. Security-related XML methods

XML is a cornerstone of Web service standards. It forms the basis of safe Web services. Traditional work for reliability including XML document encryption, data integrity testing, and sender confirmation is a relatively simple process. However, more and more documents require particular operations for certain parts according to different security levels. The security-related XML methods involve XML encryption, XML signature [15], XACL (Expanded Access Control Language), etc. XML signature and XML encryption can ensure the confidentiality, integrity, authentication, and nonrepudiation as well as the end-to-end security [16]. XML encryption and XML signature generally adopt asymmetric encryption, which can guarantee the data security. However, for massive data, these asymmetric encryptions and decryptions will take lots of computational resources, which gives rise to performance degradation [17].

2.4. Related frameworks

JGuard [13] provides easy security (authentication and authorization) in Web and other applications. It is built over the JAAS (Java Authentication and Authorization Service) framework. JAAS is a relative obsolete framework. Although its authentication function is acceptable, its authorized access module is bad and frustrating. In addition, JAAS has a close relationship with the security of the virtual machine, such as judging whether to load a class. Moreover, JGuard does not use AOP (Aspect-Oriented Programming).

Shiro [18], a project of the Apache Software Foundation, is a powerful and flexible open-source Java security framework. Its predecessor is JSecurity. Usability is the ultimate goal of this project. It can provide one-stop service for many security requirements. It has good flexibility and can work in a variety of application environments. Because of its simplicity and its powerful capacity, it has attracted much attention from developers. Even some big companies, such as Katasoft, Sonatype, and MuleSoft, apply it for Web applications. However, it does not provide access control mechanisms for domain objects, which limits its applications in some fields.

Spring Security provides powerful and flexible security solutions for enterprise applications. It is a stable and mature approach based on the Spring framework, using Spring dependency injection. Meanwhile, it provides comprehensive authorization services, which are also easily integrated with existing databases. It also provides access control of three levels that are mentioned above. However, this security architecture does not provide the security mechanism for blocking Web attacks. Although Spring Security can effectively ensure authorized access to resources, the security interceptor mechanism cannot filter those illegal access requests and prevent systems from being damaged by illegal users. Moreover, Spring Security provides an access control list (ACL) to achieve the access control of domain objects, but it needs to add a new table to manage the ACL. It usually takes a considerably long time to query this table, which reduces the efficiency of the security system.

Our research work is done to overcome the shortcomings of Spring Security mentioned above.

3. Design and implementation

The improved Web security framework presented in this paper is achieved by enhancing and extending Spring Security from three aspects: identity authentication, access control, and secure transmission. Since this security framework takes VeePalms as a representative application, the following discussion about the security framework focuses on it. However, those ways mentioned below are also suitable for many other Web service-based systems.

3.1. Architecture

Figure 1 shows the architecture of the security framework presented. According to characteristics of the Web service-based systems, the improved security framework is based on Spring Security, and some improvements are made to the safety intercept module of Spring Security. Different requests are processed distributed through the security interceptor: for login authentication requests, they will be sent to the identity authentication module; for encrypted requests, the data security transmission module will deal with them; for requests of other types, the access control manager module is in charge of them to decide whether the requests have permission to access resources [19]. The access control manager queries an ACL to judge whether the requests are legal or not. ACL maintains a mapping table about the relationship between resources and authority, through which the access control manager can estimate whether the user is authorized to access resources or not.

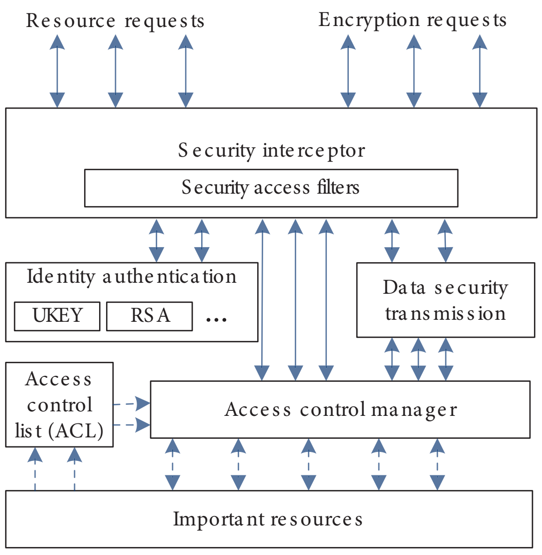


Figure 1. Security framework architecture.

The identity authentication module adds U-key identity authentication and RSA (Rivest–Shamir–Adleman) encryption authentication to improve the reliability of the authentication module of Spring Security. As shown in Figure 1, some security filters are added at the security interceptor in the Spring Security framework, so as to ensure the safety of the system resources accesses, integrating the data security transmission and identity authentication modules.

3.2. Access control model

Figure 2 illustrates the architecture of the access control model in the improved security framework. The architecture consists of three parts: resource requests from clients, requested resources, and access control policies. In this architecture, the RBAC [20] model is used to implement access control policies [21].

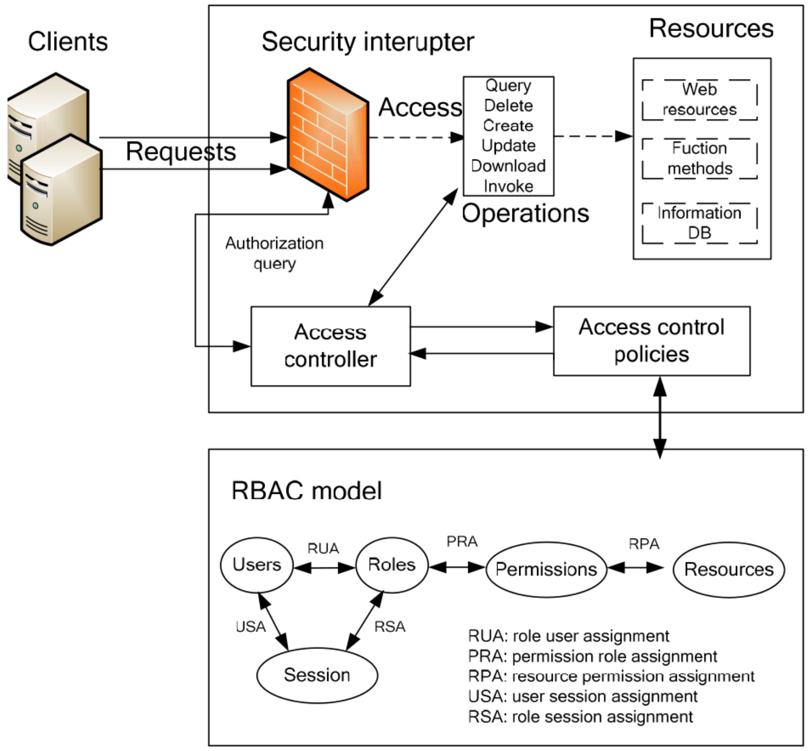


Figure 2. Architecture of access control model.

The role-based access control model consists of the following five components: users, roles, permissions, resources, and sessions. As shown in Figure 2, there are also five assignment relations between components. When a user tries to access an object, the system checks the rights owned by the user and decides whether to allow the access or not.

Some abbreviations are explained as follows:

U (Users): a set of operators who can independently access information system data or other resources expressed by data. In VeePalms, users are virtual experiment designers, managers, consumers, and others. If u is used to denote a particular user, $U = \{u_1, u_2, u_3, u_4, \ldots, u_n\}$.

R (Roles): a set of participators who have the same access operations to some data with the same types. There are five roles in VeePalms, which are teacher, administrator, academic dean, student, and guest visitor. When a role is assigned to a user, its permissions to resources are also assigned to the user. Thus, a role is a medium of assigning permission rights to users. If r is used to denote a particular role, $R = \{r_1, r_2, r_3, r_4, \ldots, r_n\}$.

P (Permissions): a set of rights that are used to access resources. Permission is the right of operating resource objects. These operations include creating, updating, deleting, querying, viewing, invoking, and downloading. Thus, $P = \{$*create, update, delete, query, view, download, invoke*$\}$.

W (Resources): a set of entity objects that are various organizing forms of resources saved and accessed by users, such as a file, a page, a database record, or function methods. In VeePalms, objects are multiform experimental resources. If w is used to denote a particular resource, $W = \{w_1, w_2, w_3, w_4, \ldots, w_n\}$.

S (Sessions): a set of mappings between users and roles. When a user activates a role, a session is created. A session is a unit of access control. Each session is associated with a single user, and each user is associated with one or more sessions. Sessions and roles are in a many-to-many relationship.

In RBAC, key relations are related to role, user, and permission. Two important relations with them are shown as follows:

RUA (Role User Assignment): a many-to-many mapping of the role-to-user assignment relation. $RUA = R \times U$.

PRA (Permission Role Assignment): a many-to-many mapping of the permission-to-role assignment relation. $PRA = P \times R$.

RPA (Resource and Permission Assignment): a many-to-many mapping of the resource-to-permission. $RPA = R \times P$.

The access control in this model is determined by roles, permissions, and resources. After RUA, PRA, and RPA are completed, users can access resources appointed according to corresponding permissions. The implementation of the mechanism is introduced in the next section.

3.3. Strategy and implementation

A safe Web service system should meet five security requirements: authentication, authorization, confidentiality, integrity, and nonrepudiation. Those five requirements fall into three parts according to their actions: the authentication mechanism used for guaranteeing the authorization, the authorized access mechanism for ensuring the authentication, and the secure transfer of data, which ensures confidentiality, integrity, and nonrepudiation. To sum up, in order to ensure the security of a Web service, there are three security modules need to be achieved, which are the identity authentication module, the access control module, and the data security transmission module.

The identity authentication module discussed in this paper improves the authentication module of Spring Security by combining U-Key authentication and the RSA encryption mechanism. For a U-key user, the user's CA certificate is stored in U-key hardware. When the user uses the U-key to login, the authentication information is sent to a certificate server, after being encrypted with the private key. Its security authentication module decrypts the information to get the user's identity. For a non-U-key user, the system adopts RSA encryption to ensure the safety of their authentications. All the above are combined with the authentication module of Spring Security.

The access control module is also built based on the Spring Security framework. Although the identity authentication mentioned above can guarantee users' legitimacy, it is not able to ensure authorized accesses to resources yet. Here we construct an access control module by improving the corresponding modular part of the Spring Security framework, especially in optimizing the domain object-level access control.

The data security transmission module encrypts users' core data selectively, adopting a new hybrid encryption mechanism by combining the RSA and DES (Data Encryption Algorithm). The server-side maintains a pair of RSA keys, which are the private key and public key, and transmits the public key messages back to the client side. Then the client side generates the DES key, which is used to encrypt vital data, and the RSA public key is applied to encrypt the DES keys.

In the following, we discuss the identity authentication, the access control, and security transmission in detail.

3.3.1. The identity authentication

The Spring Security framework has its own set of identity authentication methods, such as HTTP basic authentication, digest authentication, OpenID authentication, and form authentication. However, all these authentication methods cannot meet the security requirements of the Web services-based systems enough.

Different from these existing methods, we combine the U-key method and RSA encryption with the Spring Security framework to ensure the identity authentication. The two identity authentication methods have their own characteristics.

The Springer Security framework has provided interfaces for users to add their own authentication components.

To be compatible with other systems, Spring Security provides an *AuthenticationProvider* interface. This interface can meet different needs of different identity authentication systems. Moreover, it can guarantee the seamless integration of the identity authentication and the authorization modules. The relationships of the classes in the identity authentication of Spring Security are shown in Figure 3.

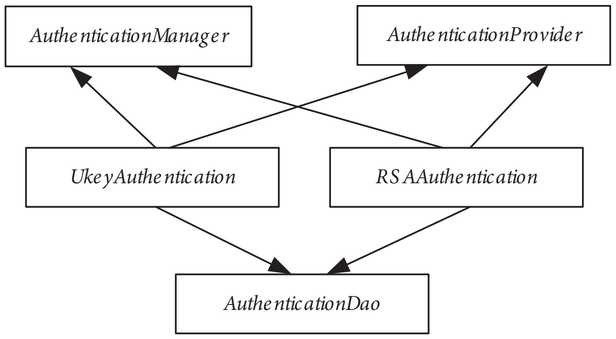


Figure 3. The relationships of the classes in the identity authentication.

Here, *AuthenticationManager* is for the authentication management. Any login requests will be committed to it to deal with. It will select corresponding authentication methods according to the coming requests. *UkeyAuthentication* and *RSAAuthentication* are the two authentication classes that realize the concrete authentication methods of *U-key* and *RSA* encryption presented in this paper, by implementing the *AuthenticationManager* interface. The authenticate function in the *AuthenticationManager* is provided for *AuthenticationProvider* to complete the user authentication request. As developers, we just need to implement two interfaces of *AuthenticationManager* and *AuthenticationProvider* to realize our own authentication methods in the Spring Security framework.

More about the U-key method and RSA encryption for identity authentication are given as follows.

3.3.1.1. Identity authentication based on U-key

The identity authentication module adopts the U-key authentication mechanism, which is implemented by a public-key infrastructure (PKI). The PKI provides a mechanism to support both identity certificates and access control. The core of the PKI is that of the certification authority (CA) that mainly deals with key pairs (a private and a corresponding public key) issuing. The private key must remain secret, under the control of its owner, while the public key must become available to anyone who wishes to have some type of transactions with the owner of the private key. A typical PKI consists of PKI policy, software and hardware, CA, RA (Register Authority), and PKI application. The purpose of the CA is to ensure the digital certificates to be used by other parties. The jobs of the RA are to process users' requests, confirm their identities, and induct them into the user database.

The user's CA certificate, which identifies only the identity of users, is stored in the U-key hardware. The authentication information is encrypted by private key and sent to the authentication server, in which the security authentication module decrypts the identity of the user by public key. The public keys of users should be under unified management by the authentication server. After the authentication succeeds, the U-key owner is granted the permissions assigned to his/her role.

In the identity authentication based on the U-key, the main function of the server is to verify the identity of the U-key. There are two steps in this process: verifying the validity of the user certificate in the U-key and verifying the validity of the private key of the U-key. Figure 4 shows the authentication process of U-key.

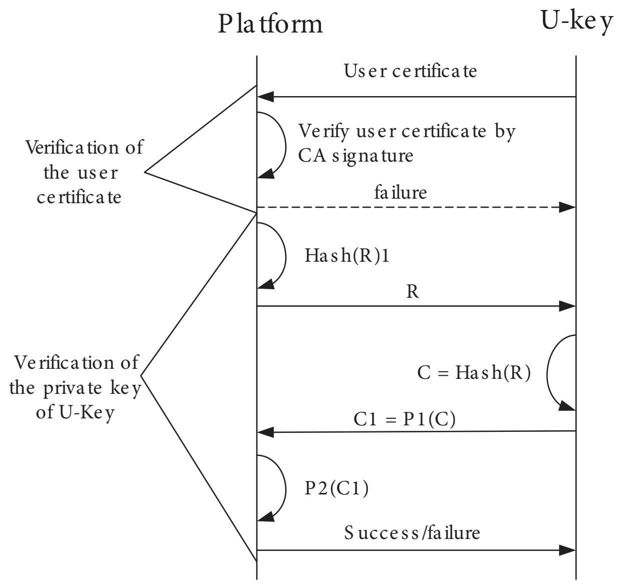

Platform
U-key
User certificate
Verify user certificate by CA signature
Verification of the user certificate
failure
Hash(R)1
R
Verification of the private key of U-Key
C = Hash(R)
C1 = P1(C)
P2(C1)
Success/failure

Figure 4. Process of U-key authentication.

The premise of the first step is that the server has already had a dependable CA certificate. The process of the steps is listed as follow.

Step1: The server receives the user certificate from the U-key client.

Step2: The server gets the public key from the user certificate and decrypts the CA signature part of the user certificate, which is named Digest-A, while the server digests the other part of user certificate, which can be named Digest-B.

Step3: Compare Digest-A with Digest-B. If Digest-A equals Digest-B, the user certificate is issued by the CA. Otherwise, the authentication fails.

Due to the possibility of falsifying the user certificate, verifying the validity of the user's certificate is not enough for security. It also requires that the client should match with the certificate provided by the user. Because the private key of the user is exclusive, the user's identity can be confirmed by digital signature.

The premise of the second step is that the server has already verified the validity of the user certificate, which is issued by the CA. The process of the second step is described in detail as follows.

Step1: The server generates a random string R, and deals with it by hash function (here, MD5 is applied), named Hash*(R)1.*

Step2: The server sends R to the client, and the client also processes R by hash function, named C. Then the U-key encrypts C with a private key, named *C1.* Then *C1* is sent to the server.

Step3: The server receives *C1* and decrypts it with the public key of the user certificate, called Hash*(R)2.* Compare Hash*(R)1* with Hash*(R)2.* If Hash*(R)1* equals Hash*(R)2,* the U-key authentication is successful; otherwise, it fails.

3.3.1.2. Identity authentication based on RSA encryption

Usually, there are multirole users in this kind of system. Due to some special reasons such as huge scale of systems, operating costs, and levels of users, it is impossible or unreasonable to ensure that all users have U-keys. The U-key authentication is used only for logins of important roles, such as teachers, academic deans, and administrators. Meanwhile, other types of users, such as students and guests, need another way to ensure security authentication.

For the non-U-key users, the system adopts RSA encryption to ensure the safety of their authentications. The RSA encryption algorithm is an asymmetric cryptographic algorithm, widely used in public key cryptography standards and electronic businesses.

When someone first visits the system, the system generates a pair of RSA public and private keys, which are stored in the session of the request, and sends the RSA public key to the client. Then the client logs in, using the password and account encrypted by the RSA public key. Meanwhile, it sends the encrypted cipher text to the server. After receiving the login request, the server searches for the RSA private key from the session and decrypts the password by RSA algorithm. Even if this kind of cipher text encrypted by RSA is wiretapped through the Internet, it is hard to decrypt it since the public key is required, and it will become more difficult to decrypt with RSA key lengthening. Figure 5 shows the procedure of the process.

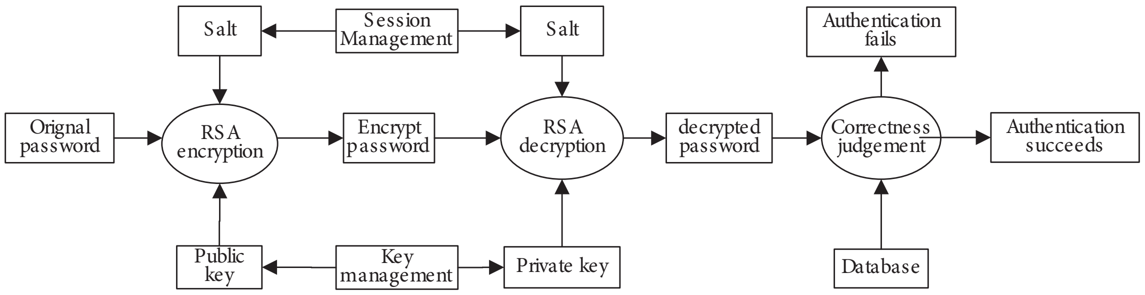


Figure 5. The procedure of the identity authentication based on RSA.

3.3.2. Security access control

3.3.2.1. Authorized access

Although the identity authentication can guarantee users' legitimacy, it cannot ensure authorized accesses to resources. The access control module of the Spring Security framework is a suitable choice to control the authorized accesses of users, which has excellent expansibility and portability.

As shown in Figure 6, in our security framework, Spring Security and the group frame SSH (Structs+Spring+Hibernate) are integrated.

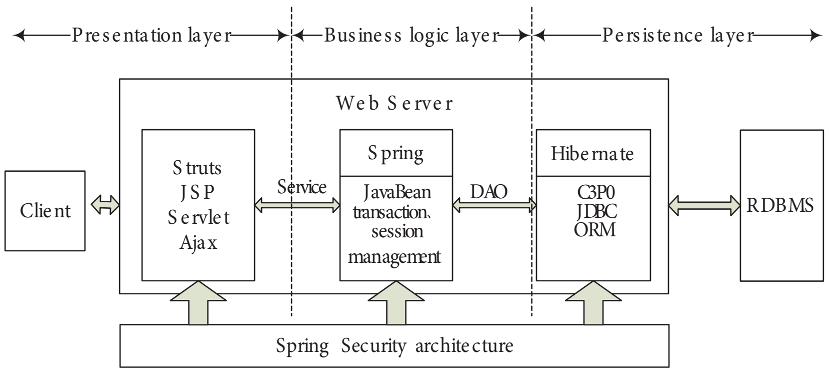


Figure 6. Integration of Spring Security and SSH.

Spring Security achieves three levels of access controls: URL-level access control, method-level access control, and domain object-level access control, which correspond to the presentation layer, business logic layer, and the persistence layer, respectively.

Users usually visit system sources through HTTP protocol. The HTTP request is the basic unit of the access control target. It is made up of following components:

Request Line: It is composed of request method, request URI (Uniform Resource Identifier), and HTTP version, which are separated by a blank space, such as "GET /news.asp HTTP/1.1".

Message Header: It is composed of domain name and value pairs, which are separated by a colon. For example, "Host: http://dome.com:80" expresses the host and port of the requested resource.

Entity Body: It carries the data related to request, such as some potentially needed parameters. It is indicated by Content-Length or Transfer-Encoding. Content-Type expresses the type of transport data.

After a user is successfully authenticated by the server, the server grants all permissions of the role to the user. The permissions assigned to the user are stored in the global session of the user by the form of key-value: <*"grantedauthority", authorities*>. The key is named *grantedauthority* and the value is a list of authorities: $authorities = \{p_1, p_2, p_3, \ldots, p_n\}$. Thus, it is unnecessary to query the database to get the permissions of the user again. Moreover, once the permissions are assigned to the user, they cannot be changed until the user logs out. Not all permissions are stored in the session. In most cases, just the identifier of permission is recorded, which can reduce memory space of the client effectively.

3.3.2.1.1. URL-level access control

What the URL-level access control achieves is to make users access URL resources safely, such as a picture, a JSP page, or a XML file. The domain name of the Message Header takes the information of the requested resource, such as "http://ip:80/beginExperiment.jsp, http://ip:80/sport.xml".

3.3.2.1.2. Method-level access control

During the response process for a user's request, it is necessary to call the business logic method under safety control. At this moment, the safety interceptor of Spring Security is triggered, which estimates whether the user has the authority to call this method.

Spring Security takes the thought of AOP, which has weak coupling with system integration. It can be realized by the mechanism of Java annotation. There are four comments: *@PreAuthorize*, *@PreFilter*, *@PostAuthorize*, and *@PostFilter*. They can be invoked by global-method-security namespace elements. For example:

@PreAuthorize("hasPermission('Create_ User')")

public void create(User user);

This means that only the user granted the permission *Create_ User* can invoke the method above. *@PreAuthorize* is the most frequently used comment.

3.3.2.1.3. Domain object-level access control

For the domain object-level access control, Spring Security achieves it through access control lists (ACLs), by which users can modify and delete specified data lines. However, the line-level access control of Spring Security brings some negative influences for the system performance. It needs to create a new record in the ACL for every record to be protected. When the record protected is accessed, the ACL should be queried and the new record should be checked every time to judge whether the request has the right or not. This procedure reduces the performance of the system largely.

Here we present an improved approach to enhance the efficiency of the system. According to the system's requirements, normally, not all but some of the tables need to be protected. In fact, these tables all have the field of owner that indicates the owner of the corresponding resource, so we can judge whether the user is the owner of some line of data owners to achieve line-level data access control. The basic information of the users can be saved in the session during the period of user identity authentication. When the information is required, only the information parameters of the user session are needed to be transferred. Thus, it is unnecessary to query ACLs every time, which reduces the overhead of the system largely. Figure 7 shows the architecture of the domain object-level access control.

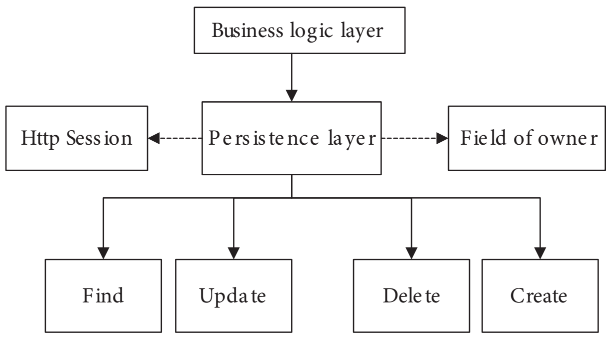


Figure 7. Architecture of the domain object-level access control.

The ACL just stores the names of the table that need to be protected. When some method of the business logic layer needs to call the interfaces of the persistence layer, if the table accessed is a protected table, the user information in the session will be transferred as a function parameter. Usually, the persistence layer just needs to realize four functions, including *update*, *delete*, *create*, and *find*. These functions should add two more parameters, which are the *HttpSession* of the user and the *owner* field. When the persistence layer calls these function, it can be judged whether the user is the data owner or not, and it is decided whether the user has the access right to resources or not correspondingly.

3.3.2.2. Security filter chain

The Spring Security framework only focuses on banning unauthorized resource accesses. Although it can ensure authorized accesses to resources effectively, it cannot solve Web attacks such as SQL injections, cross-site scripting, or insecure HTTP methods from illegal users and users with partial rights. Its security interceptor mechanism cannot get good performance on filtering those illegal access requests. Therefore, a series of security access filters for the system are desired to be added to intercept the illegal access requests and fix the high-severity security bugs, in order to improve the system security.

The U-key authentication can ensure the safety of authentications of important roles. After being authenticated, users can get what they want through HTTP requests. Therefore, it is necessary to make a filter chain to filter illegal requests by detecting every part of the HTTP requests, including Request Line, Message Header, and Entity Body.

Figure 8 shows the improvement of Spring Security access control by adding a series of security access filters. The HTTP method filter can filter out illegal accesses brought by Request Line, such as *delete*, *move*, *search*, and *copy*. Meanwhile, the invalid character filter can filter out requests containing illegal visiting characters in the parameters of Entity Body, such as *script*, *alert*, *window*, *open*, and *where*, which can avoid SQL injection attacks, cross-site scripting, and other Web attacks effectively. Other filters, such as an invalid session filter, have been also realized for preventing the system from other illegal accesses.

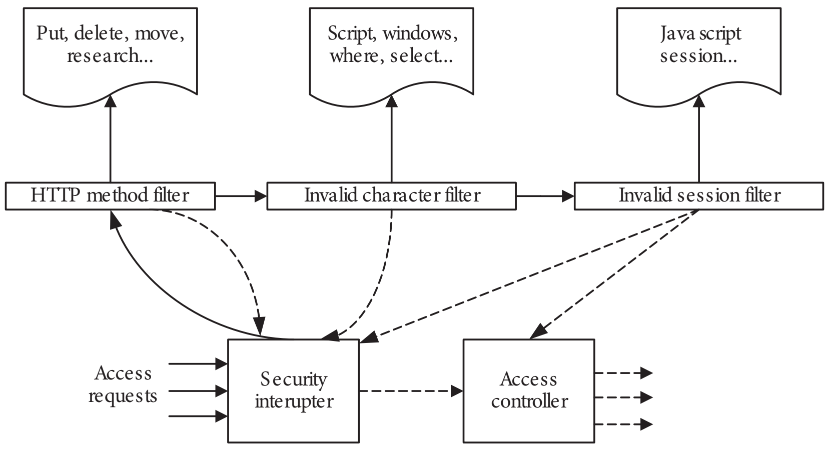


Figure 8. Process of access control of Spring Security.

3.3.3. Transmission security

The data security transmission module adopts XML encryption and XML signature technology. This transmission security module is just used when the users invoke some important interfaces of the Web service, and the transmission is based on SOAP messages. SOAP messages are described by XML, so they can be dealt with by XML encryption and XML signature.

Considering the big overhead of encryption performance of asymmetric encryptions, our system combines asymmetric cryptography RSA with the traditional symmetrical encryption algorithm DES to balance the overhead and security level of the system, by referring to the PGP (Pretty Good Privacy) e-mail encryption standard [22]. Since the speed of DES encryption is much faster than that of RSA, the DES algorithm is applied for the encryption process, which uses a random approach to generate the private key with the encryption of explicit messages at first. Then the key is encrypted by RSA algorithms. In this way, the recipient also uses RSA to decrypt this random key and then decrypts messages by DES with the key decrypted. This kind of procedure of encryption and decryption can take advantage of the confidentiality of asymmetric cryptography and the promptness of symmetrical encryption. The encrypting and decrypting process based on RSA and DES is shown in Figure 9.

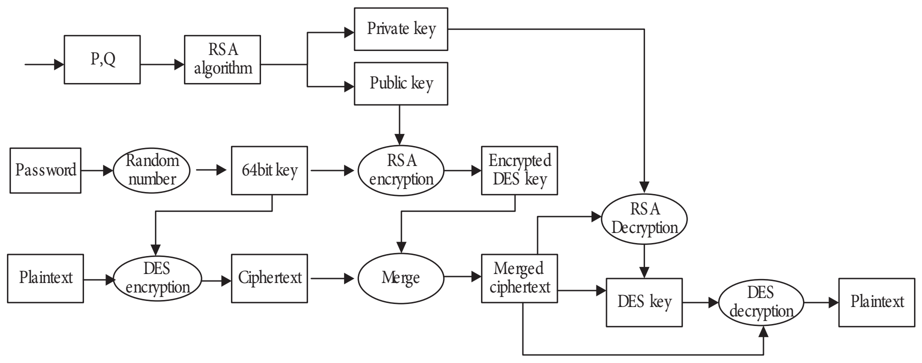


Figure 9. The process of encrypting and decrypting of RSA and DES.

First, two large prime numbers P and Q can be obtained through a prime generation algorithm. Then the key generation algorithm of the RSA encryption algorithm should be applied to generate a RSA public key, which will be released in a certain way, and a private key for preservation. Next, a 64-bit random number used as the DES session key is generated through a linear congruential method for DES encryption and decryption. The RSA public key is used to encrypt the session key that will be encrypted, saved, and combined with the DES encrypted cipher text. Finally, the receiving side decrypts the sent cipher text with the private key generated by RSA.

To make the enhancements of the presented security framework clearer compared with the traditional Spring Security framework, we show the improvements in Table 1.

Table 1. Comparisons of Spring Security and the improved security framework.

		Spring Security	Improved framework
Safe authentication		Based on account and password with HTTP basic authentication.	Based on a combination of U-key method and a RSA algorithm.
Security access control	Authorized access	RBAC model.	Optimizing the domain object-level access control of RBAC by reducing ACL access.
	Security interceptor	Can ensure authorized accesses; however, cannot solve Web attacks from illegal users and users with partial rights.	A series of security filters are added to keep Web attacks such as SQL injections, cross-site scripting, and insecure HTTP methods away.
Secure transmission		No special specification. DES is often used for its efficiency.	Combining RSA and DES to balance the overhead and security level.

4. Performance evaluation

To approve the effectiveness of the framework presented in this paper, we test it in VeePalms by adopting IBM Rational AppScan [23].

Rational AppScan is a leading security testing tool in the information industry. It can provide security flaw scanning, safety reports, fix advices, etc. It is a kind of black-box testing tool, which means that the testers do not need to know about the structure of the system under test. Rational AppScan has a very large and complete feature library of security flaws. The testing method is to insert various application attacks in the feature database into HTTP requests, then analyze the HTTP responses to determine whether there are corresponding system security flaws.

First, AppScan does an Explore process (its behavior is similar to Web crawlers) for the system. It explores URLs of the system iteratively from some URL and stops until all the pages have been explored or the maximum number of pages to be explored is reached. At the same time, AppScan analyzes each page obtained and determines whether it needs to be tested. Then, according to their own safety rules, it constantly tests the links required to be tested.

The working framework of Rational AppScan is shown in Figure 10.

The VeePalms architecture is shown in Figure 11. The system is deployed in a high-performance cluster. Its nodes are divided into four categories. Additionally, a test server is added for IBM Rational AppScan. The details are shown in Table 2.

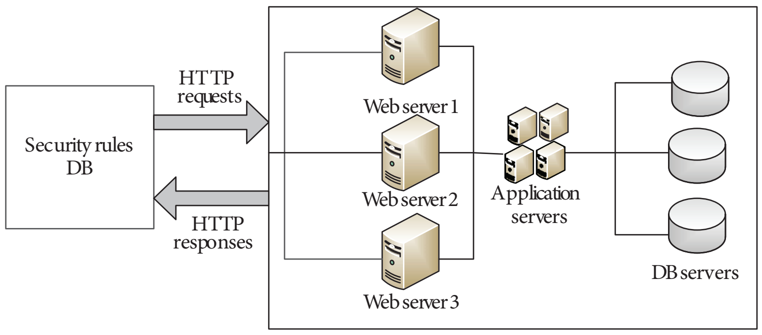


Figure 10. The working framework of Rational AppScan.

Table 2. Servers and parameters of VeePalms.

Servers	Parameters	Type
Web servers	CPU E5520, Mem. 16 GB, HD 2 × 146 GB SAS	NF5220
DB servers	CPU E5520, Mem. 32 GB, HD 2 × 300 GB SAS	NF560D2
Center management server	CPU E7450, Mem. 16 GB, HD 2 × 146 GB SAS	NF5120
Computing nodes and test server	CPU E5520, Mem. 16 GB, HD 2 × 146 GB SAS	NF5120

Here, the center management server and computer nodes are in charge of virtual experiment simulations, which are not related to data security of the improved security framework. We thus focus on the Web servers and DB servers. The software configurations of these servers for testing are shown in Table 3.

These servers are in gigabit Ethernet. Web servers use the open source Apache-Tomcat 6.0.24. Web applications apply Struts, Spring, Hibernate, and the Spring Security framework. The Oracle database and a clustered MongoDB system [6] are used for structure and unstructured data storage, respectively.

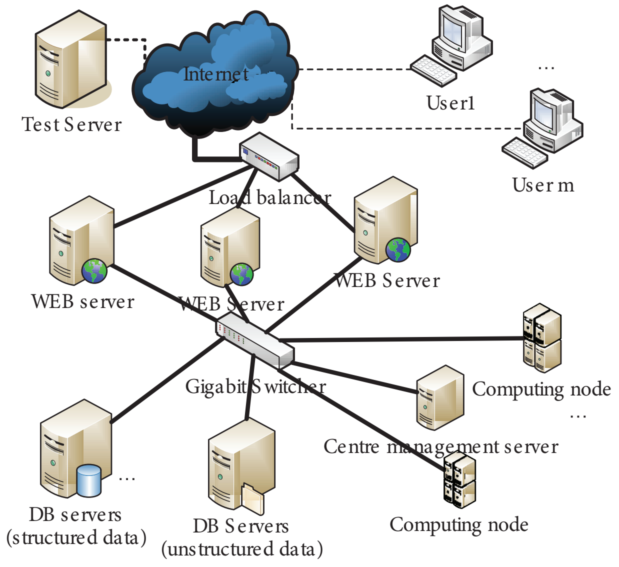


Figure 11. Architecture of VeePalms.

Table 3. Software environments of the servers for testing.

Servers	Software name	Version
Web servers	Tomcat	6.0.24
	Struts	2.1.6
	Spring	3.0.1
	Hibernate	3.0
	Spring Security	3.0.2
	C3P0	0.9.1
	Axis	1.4.1
	Java	1.6.0
DB servers	Oracle and clustered MongoDB	11g
Test server	Rational AppScan	7.8.1

The data in the virtual experiment platform are in multiple forms, such as experimental components (XML), experimental guidelines (video, Word), lists of scores (TXT), experimental scenes (XML), and simulation source files (MO).

In the platform, we realize both the traditional Spring Security and the improved security framework presented in this paper.

First, we make comparisons of encryption performances of DES, RSA, and DES&RSA. The selected four files have different sizes: 10 KB, 30 KB, 60 KB, and 100 KB. The detailed encryption time is shown in Table 4.

Table 4. Time of encryption by DES, RSA, and DES&RSA.

Size / Algorithm	10 KB	30 KB	60 KB	100 KB
DES (Spring Security)	4 ms	10 ms	19 ms	35 ms
RSA	1209 ms	3542 ms	7080 ms	11,744 ms
DES&RSA (the improved security framework)	96 ms	111 ms	160 ms	586 ms

As seen in Table 4, although DES&RSA has more time in encryption compared with DES, it has an obvious time advantage compared with RSA. At the same time, DES&RSA can provide a securer guarantee than DES.

Second, Rational AppScan is used to test the security holes of the system with the traditional Spring Security and the ones with the improved security framework presented in this paper. There are five roles in VeePalms, which are administrator, academic dean, teacher, student, and guest visitor. The security test results of every user space are shown in Table 5. It is obvious that the number of the holes in the system after using the improved security framework is reduced largely. Table 6 shows the reduction ratios of holes in detail. Compared with traditional Spring Security, after applying the improved security framework, the number of security holes in every user space is reduced by more than 60%. Of course, since a series of security filters are added, more time is needed for the access of URLs. Table 6 also shows the average increase ratios of access time for different roles. The URL access time with the improved security framework increases a little compared with Spring Security. Generally, these slight time increases are acceptable taking the advantages into account. The ratios of security holes of different user spaces are shown in Figure 12. The comparison results of ratios of different severity levels are shown in Figure 13. For convenience, the names of roles are written in abbreviations: administrator as Adm., academic dean as Acd., teacher as Tea., student as Stu., and guest visitor as Vis.

Table 5. Comparison of numbers of security holes.

Role	Sum. of URLs scanned	Num. of security holes	
		With Spring With Spring	With the improved security framework
Adm.	89	49	17
Acd.	24	17	4
Tea.	42	15	3
Stu.	20	17	3
Vis.	30	19	4

Table 6. Comparison of ratios of security holes and URL average access time.

Role	Ratio of security holes		Reduction ratio of holes	Increased ratio of access time
	With Spring Security	With the improved security framework		
Adm.	41%	14%	65.9%	12%
Acd.	31%	9%	72.7%	10%
Tea.	23%	5%	78.3%	14%
Stu.	30%	7%	78.8%	11%
Vis.	29%	11%	62.0%	7%

As seen in Figure 13, compared with traditional Spring Security, after applying the improved security framework, all security holes with high severity in the system have been fixed and medium and low severity holes have decreased dramatically. In detail, the presented Web security framework can solve most systematic bugs of extremely high severity, such as incomplete account blockade and cross-site scripting establishment. Moreover, it also solves lots of bugs of medium severity, including enabling unsafe HTTP method, decrypted login request, and so forth. Figure 14a shows the details of security holes in every role with the traditional Spring Security, and Figure 14b shows ones with the improved security framework.

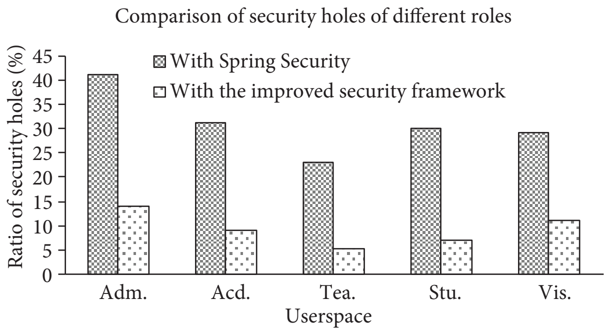


Figure 12. Ratios of security holes of different user spaces.

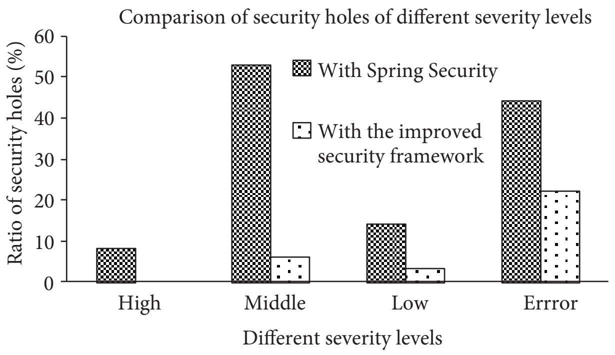
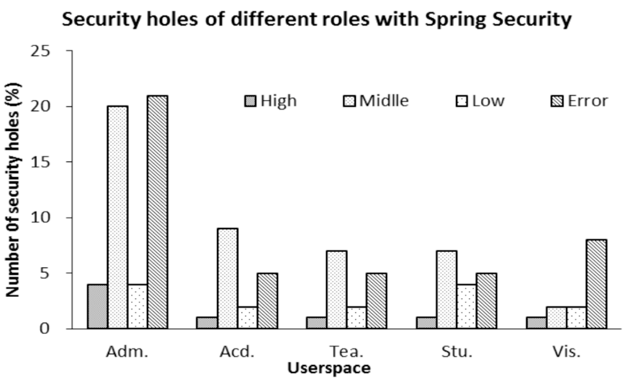
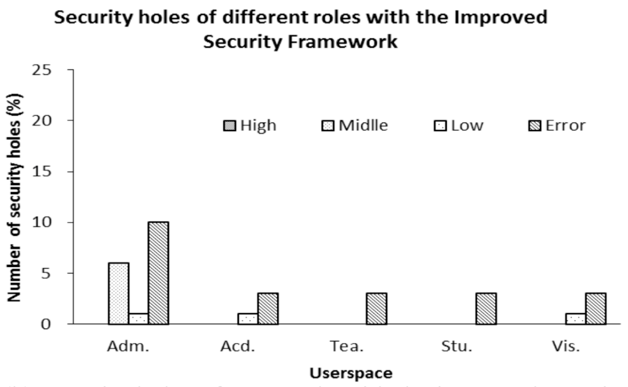


Figure 13. Test results of different severity levels.

VeePalms with the improved security framework has run for 2 years. Figure 15 shows the portal and experiments of VeePalms. More than 180 experiments from 27 courses have been developed and integrated into the platform. These courses are from five disciplines, which are computer science, mechanism engineering, control engineering, electrics and electronics, and middle school science.



(a) Security holes of every role with Spring Security



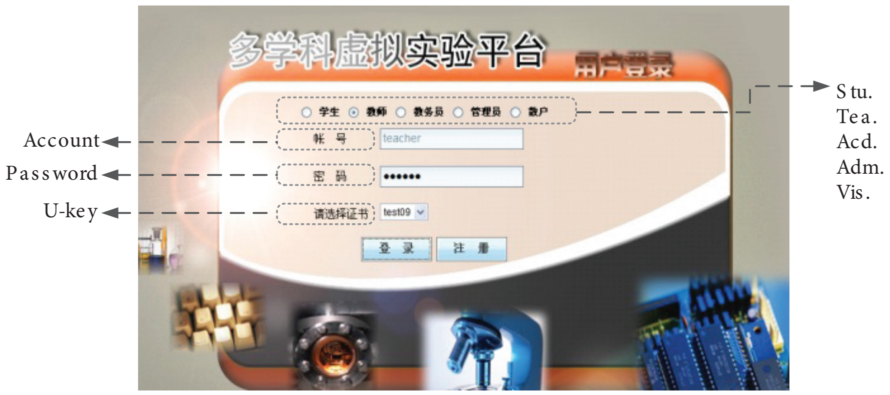
(b) Security holes of every role with the improved security framework

Figure 14. Details of security holes in every role.

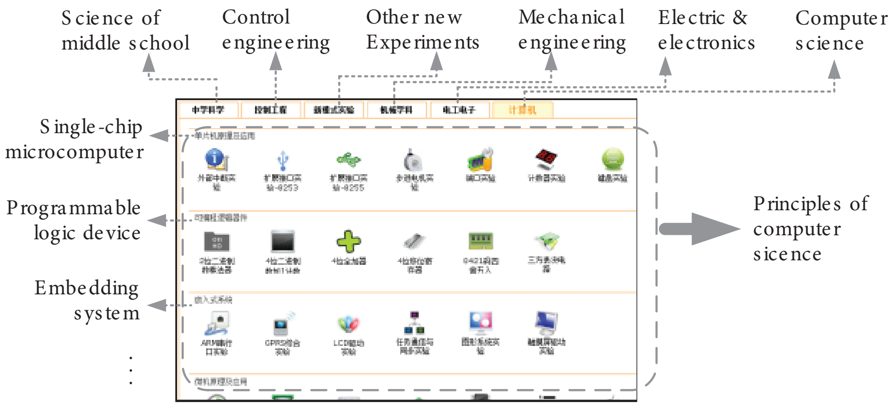
Through cooperation with the National Center for Open & Distance Education, a leading long-distance e-learning organization in China, VeePalms has provided experiment services for its more than one million students from all over the country. The practice proves that the improved security framework can ensure the security of the system.

5. Conclusion

This paper presents a new Web security framework for Web service-based systems, which ensures the security requirements of the systems from identity authentication, access control, and safety transmission. The highly safe authentication is based on the integration of an improved authentication module of Spring Security with a U-key method and a RSA algorithm. For the authorized access, the Spring Security's ACL mechanism is improved by optimizing the domain object-level access control. For secure transmission, a compromising method is presented to take both the security level and the speed of data transmission into account by means of mixing RSA and the DES algorithm. In addition, the security interceptor of Spring Security is extended and a series of security filters are added to keep Web attacks away. The experimental data show that this security



(a) The portal of VeePalms



(2) Lists of experiments from different princples

Figure 15. The portal and experiments of VeePalms.

framework can fix all kinds of bugs with high severity existing in the Web service-based applications and can decrease the numbers of Web security bugs with medium and low severity effectively. The safety of the Web system has been improved obviously. Moreover, the effectiveness of the improved security framework has been proved by practice.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China under Grant No. 60903173 and by the projects of the National Science & Technology Pillar Program of China under Grant No. 2012BAH14F02 and No. 2008BAH29B00.

References

[1] Roberts-Morpeth P, Ellman J. Some security issues for Web based frameworks. In: 7th IEEE and IET International Symposium on Communication Systems Networks and Digital Signal Processing; 21–23 July 2010; Newcastle upon Tyne, UK. Piscataway, NJ, USA: IEEE. pp. 726-731.

[2] Xie W, Ma H. A policy-based security model for Web system. In: 2003 International Conference on Communication Technology; 9–11 April 2003; Beijing, China. Beijing, China: IEEE. pp. 187-191.

[3] Peng S, Han Z. Trust of user using U-key on trusted platform. In: 8th International Conference on Signal Processing; 16–20 November 2006; Beijing, China. Piscataway, NJ, USA: IEEE. pp. 3023-3026.

[4] Jiang W, Li H, Jin H, Zhang L, Peng Y. VESS: An unstructured data-oriented storage system for multi-disciplined virtual experiment platform. In: 4th International Conference on Human-Centric Computing; 11–13 August 2011; Enshi, China. Heidelberg, Germany: Springer Verlag. pp. 187-198.

[5] Jiang W, Jin H, Yu J. SOA-based virtual experiment education research. Communications of the CCF 2010; 6: 64-69 (in Chinese with abstract in English).

[6] Jiang W, Zhang L, Qiang W, Jin H., Peng Y. MyStore: A high available distributed storage system for unstructured data. In: 14th IEEE International Conference on High Performance Computing and Communications; 25–27 June 2012; Liverpool, UK. Los Alamitos, CA, USA: IEEE. pp. 25-27.

[7] Field JP, Graham SG, Maguire T. A framework for obligation fulfillment in REST services. In: 2nd International Workshop on RESTful Design; 28 March 2011; Hyderabad, India. New York, NY, USA: ACM. pp. 59-66.

[8] Li Z, Xi ZW. Integration with JavaEE framework to build tourism e-business system. Adv Intell Soft Comput 2012; 163: 197-204.

[9] Ma K, Fang C. A security extension framework based on SOAP header. J Inf Comput Sci 2012; 9: 5249-5256.

[10] Park EJ, Kim HK, Lee RY. Web service security model using CBD architecture. In: Fifth ACIS International Conference on Software Engineering Research, Management, and Applications; 20–22 August 2007; Busan, South Korea. Piscataway, NJ, USA: IEEE. pp. 346-350.

[11] Niu Y, Wu L, Zhang X. An IPSec accelerator design for a 10Gbps in-line security network processor. J Comput 2013; 8: 319-325.

[12] Chess B, McGraw G. Static analysis for security. IEEE Secur Priv 2004; 2: 76-79.

[13] Prunicki A, Elrad T. Aclamate: An AOSD security framework for access control. In: 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing; 29 September–1 October 2006; Indianapolis, IN, USA. Los Alamitos, CA, USA: IEEE. pp. 293-300.

[14] Yu D. Role and task-based access control model for web service integration. J Comput Inf Sys 2012; 8: 2681-2689.

[15] Chen Y, Guo W, Zhao X. Study of XML digital signature for resource document fragment. In: 2nd International Conference on Information Science and Engineering; 4–6 December 2010; Hangzhou, China. Piscataway, NJ, USA: IEEE. pp. 1541-1544.

[16] Xiao Z, Yang Y, Zhang W. XML-based information security technology study. In: 2nd International Conference on Software Technology and Engineering; 3–5 October 2010; San Juan, Puerto Rico. Piscataway, NJ, USA: IEEE. pp. 236-239.

[17] Nordbotten NA. XML and Web services security standards. IEEE Commun Surv Tut 2009; 11: 4-21.

[18] Garrison WC 3rd, Lee AJ, Hinrichs TL. The need for application-aware access control evaluation. In: Proceedings of the New Security Paradigms Workshop; 18–21 September 2012; Bertinoro, Italy. New York, NY, USA: ACM. pp. 115-125.

[19] Sidharth N, Liu J. IAPF: A framework for enhancing web services security. In: 31st Annual International Computer Software and Applications Conference; 23–27 July 2007; Beijing, China. Los Alamitos, CA, USA: IEEE. pp. 23-30.

[20] Zhu M, Yan L, Ji L. An authority control model based on RBAC. In: 2011 International Conference on Computer Science and Service System; 27–29 June 2011; Nanjing, China. Piscataway, NJ, USA: IEEE. pp. 1063-1065.

[21] Oh S, Park S. Task-role-based access control model. Inform Syst 2003; 28: 533-562.

[22] Sobh TS, Amer MI. PGP modification for securing digital envelope mail using COM+ and Web services. Int J Netw Secur 2011; 13: 79-91.

[23] Tang K, Chen S, Levy D, Zic J, Yan B. A performance evaluation of Web services security. In: 10th IEEE International Enterprise Distributed Object Computing Conference; 16–20 October 2006; Hong Kong. Los Alamitos, CA, USA: IEEE. pp. 67-74.