Framework of Web Applications for Protection against Illegal Access

● Satoru Torii ● Yoshiki Higashikado ● Takayoshi Kurita

(Manuscript received December 15, 2003)

The use of Web-based application servers has recently expanded worldwide. Consequently, there is a growing threat of illegal computer access to applications. Security measures are now being implemented for applications that incorporate security logic in addition to business logic. This paper introduces a framework that protects against illegal computer access to application servers, thereby minimizing the corrections that must be made on the application side and protecting against illegal computer access.

1. Introduction

Recently constructed Web-based business application systems are being widely used in Internet services such as e-Business. These Web services (known as e-Businesses) use protocols such as HTTP and HTTPS, which usually pass through a firewall as is. For this reason, an illegal HTTP request will directly reach a Web service application. Therefore, measures are needed to prevent illegal access to Web service applications.

Simple Web application services developed using Common Gateway Interface (CGI) technology were used in the past. However, because the content of Web services has become increasingly complex in recent years, Web application systems are now being developed with Java technology based on the application server. Therefore, a Web application developer must also incorporate security logic within applications, along with more advanced program logic to accommodate the advances being made in the content of Web services.

However, a high level of skill is needed to implement security logic. Consequently, when considering the person-hours needed to develop and standardize the business and security logic, measures must be taken to accommodate both functions.

It is also necessary to 1) implement security measures on the framework side of an application server to accommodate improvements made in application security and 2) prevent an increase in application person-days.

This paper mainly describes a framework used to prevent illegal access to application servers.

2. Model of an application server

We modeled a general application server as follows (**Figure 1**):

The following describes each component of this model.

1) HTTP server

This server directly receives HTTP requests passed through the firewall from a client system for relay to a Web container. HTTP servers must have an Application Programming Interface (API) for processing HTTP and HTTPS request responses.

2) Web container

A Web container is an execution environment for running JavaServer Pages (JSP) (i.e., the view

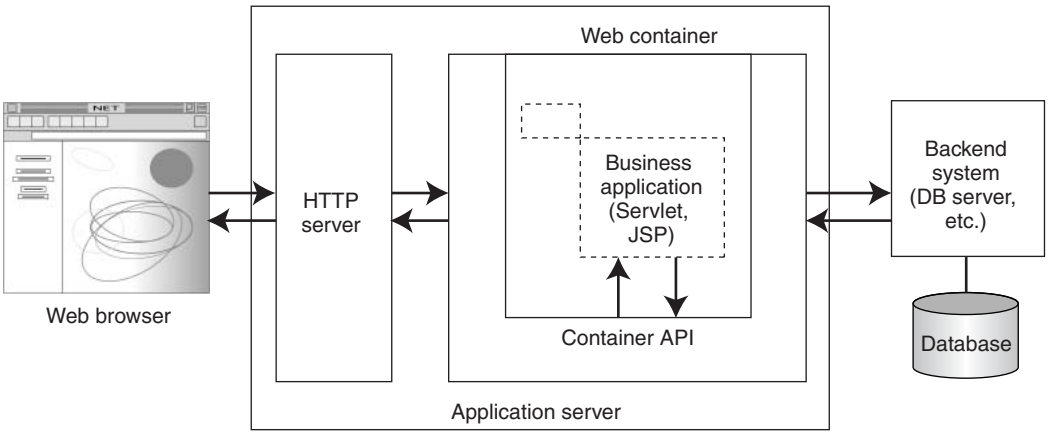

Figure 1
Model of Web application server.

of a business application) and Servlets (i.e., the main part of a business application). A Web container must include a container API for processing the contents input to and output from an HTTP server and the business application running on the Web container calling service of a Web container at execution.

3) Business application

A business application is a program that runs on a Web container that consists of JSP or Servlets. They use container APIs and connectors to call the services of containers and back-end systems (e.g., DB servers) and process their data.

3. Requirements for protection against illegal access

Generally speaking, there are three main requirements for a framework of protection against illegal access to an application server:

1) Security of each component

To ensure the security of each component of an application server, not only the components but also the links between them must be protected from illegal access.

2) Protection against illegal access

It is difficult to eliminate all vulnerability; therefore, a structure is needed to protect against illegal access and minimize any resulting damage.

3) Audit by logging surveillance

The proper performance of daily duties must be checked. If any damage results from illegal access, a function is needed to determine the extent of damage and details of the illegal access, and identify the persons responsible for the illegal access.

3.1 Components to secure

1) Securing the base software

The HTTP server is the base software of an application server, and the Web container must be secure to prevent security breaches. Therefore, an organization is needed to promote the development of safe software and establish the necessary techniques in cooperation with security experts.

2) Securing business applications

Knowledge about each business to be processed is required when constructing a business application. Until now, it was necessary to incorporate security logic into the logic of a business application, which required many person-days. Moreover, support functions that focus on implementing business logic and enabling safe construction of an application are needed.

3) Securing the entire system

All components must mutually cooperate to provide the Web application service offered by the application server. In the procedures for this mutual cooperation, breaches in security may arise and be used for illegal access.

Therefore, the level of security must be increased in the components, the links between them, and each cooperative process. Therefore, the appropriate level of security must be provided for the entire system so that consistent adjustments can be made.

3.2 Protection against illegal access

1) Checking input requests

The typical technique of illegal access is to pass a request that has been skillfully created to breach a hole in security to the application server. Such holes may exist in each component, and a variety of techniques are used to breach them. Therefore, input requests must be checked based on these factors to determine whether to receive the requests.

2) Checking output data

If an illegal access is successful, the response message might include information about internal components of the application server and personal information about a third party.

Therefore, output data must be checked to see if it contains data that must not be output to the client.

3) Protection mechanisms

If inaccurate data is detected, the protection mechanisms are immediately adjusted so they can prevent damage due to illegal access. In other words, input and output data that is determined to be inaccurate must be controlled so that it does not pass a back component or back client. Otherwise, inaccurate parts must be removed or changed so the content can be passed.

3.3 Audit by logging or surveillance

1) Surveillance of business applications

The operation of business applications may be changed due to illegal access. Moreover, it is common for two or more business applications to cooperate for processing. Therefore, the status of two or more business applications can be supervised, the surveillance results can be associated with the status of applications, and all applications can be checked to see if they are operating correctly.

2) Logging of business applications

Logging functions that record the contents of a request for information about an accessing agency or URL level are already available. However, Web pages that change dynamically in a business application according to the content of a request and do not appear in a URL are often used. Also, the processing contents of a business application are difficult to understand by using existing logging functions. Therefore, a record of the processing contents of a business application may be required for a better understanding and follow-up.

3) Securing the contents of logging

Generally, an unauthorized person may attempt to delete and alter log data without leaving any clues to such action. Moreover, when a trouble occurs in a system, a series of relevant processing histories may be requested to determine if there is a trouble in the system.

In other words, proof and authentication must be obtained, and it may be necessary to safely retain processing history data to prevent the alteration of recorded contents.

4. Framework

We have developed a framework for protecting application servers from illegal accesses that satisfies these requirements. **Figure 2** shows an image of the framework.

This framework provides a higher quality of security in each component of the application server. Moreover, it provides a higher quality of security for the entire system.

We will now describe some of the functions provided by this framework.

4.1 Separation of security logic

Conventional business applications implement both a business logic and a security logic. Therefore, the two types of logic are separated from each other. That is, the security logic is implemented on the framework side so the business

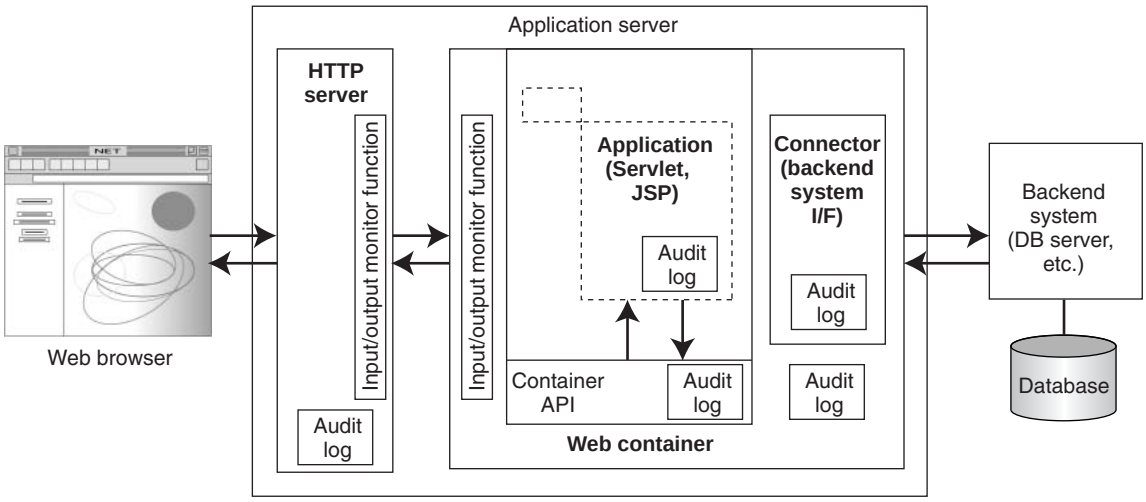

Figure 2
Framework of Web application server.

logic can focus on its tasks. As a result, when an application calls the security logic, the changes made to the business logic part can be minimized. Also, the security of not only the security logic but also common parts, for example, libraries, is improved.

4.2 Input stream surveillance and protection mechanism

The standards used to check an input stream differ according to each component, because the methods used to breach a security hole in an HTTP server generally differ from those used to breach a security hole in a business application. Therefore, our framework checks for and protects against illegal HTTP server input requests and also illegal Web container input requests.

4.3 Output stream surveillance and protection mechanism

The surveillance and control mechanism of an output stream checks and controls the validity of data output at the HTTP server, Web container, and connector level. This enables the detection of and protection against an outflow of important information such as component information, primary information being processed, and confidential information in a back-end system.

4.4 Surveillance and control mechanism for illegal business application operation

A monitoring function acquires the resource information that the operating application accesses via the connector and the screen transition information of the business application. Then, an execution surveillance mechanism checks whether the business application accessed the resource using the correct authority and whether the screen transitions are performed correctly.

If large differences in surveillance results are found based on an API call sequence defined beforehand, the call is determined to be illegal.

4.5 Integration of extraction mechanism of an audit log

The log of each component is recorded, along with information about the accessing agency and request response in an HTTP level log, the input and output to the Web container, and the container API through which the business application was called. This enables the flow of the entire business application (including components of the application server) to be audited by associating the respective records of these components.

For example, accesses can be cheeked to see if they have the proper authority by comparing

the access histories of external resources (e.g., DB servers), the history of access to the connector for database calls, and whether the correct resource was accessed.

5. Implementation

Application servers can be protected from illegal accesses by protecting their input and output.[1)]

Therefore, we developed an experimental real-time filter[2)] for HTTP server input requests and a function called Wivern[3)] for Web container input requests. These are outlined below.

5.1 Real-time filter (protection function of HTTP server)

A great deal of damage resulting from illegal accesses through security holes in standard applications such as HTTP server software and PHP has been reported. The usual response to this problem is to apply correction patches to HTTP servers. However, for certain problems, these patches are not promptly applied.

The real-time filter screens out illegal requests that may violate proper use procedures via known security holes in HTTP server software. Specifically, it looks for attacks in requests received by the HTTP server before the server processes the requests. If a request that accompanies an attack is accepted, a filter is used so the attack does not pass through the HTTP server.

Figure 3 shows the functional components of the real-time filter.

The real-time filter does the following:

1) Ensures comprehensive and accurate protection

The real-time filter sorts and checks content by preparing two or more judgment rules by which a general-purpose description is possible. It also analyzes the syntax of requested judgment parameters in a formal language and judges by considering the specification information about the server.

2) Filters without external influence

Because the real-time filter is implemented as a module of the HTTP server (even with SSL encryption of requests), it can filter content with-

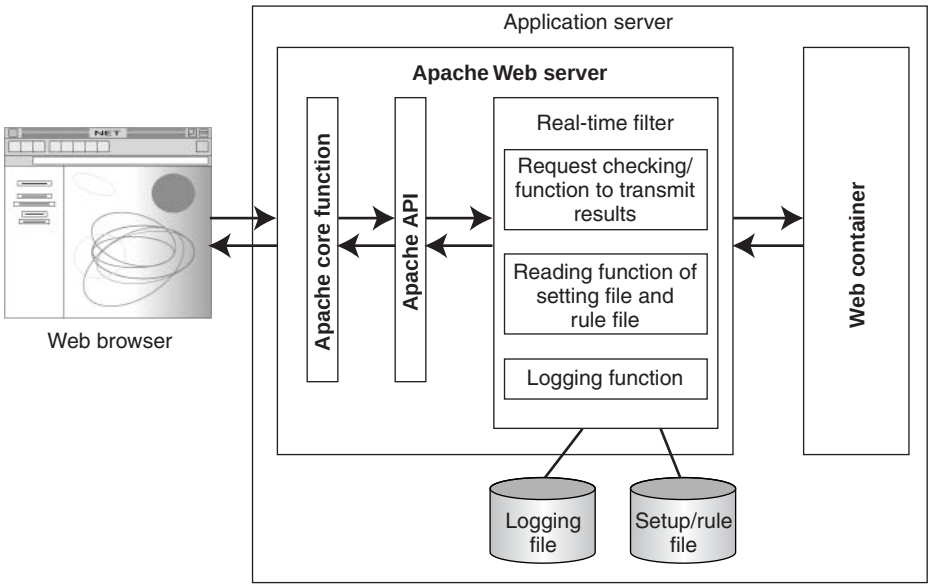

Figure 3
Real-time filter function.

out external influence.

3) Supports dynamic renewal of rules

Dynamic renewal of rules is supported, and the newest rule file can be applied without having to reboot the HTTP server in order to update the filtering rule. Therefore, it is possible to add a rule to deal with a newly discovered vulnerability while continuing service without having to stop the Web application service.

4) Temporary measures can be taken before patch application

Generally, when a security hole is discovered in a component of an application server, HTTP server, or other server, it takes time to acquire a security patch from the vendor. However, the real-time filter can be used to implement a temporary measure to close the hole until the patch is available.

5.2 Wivern protection function for Web containers

Various attacks, for example, session hijacks, cross-site-scripting (XSS), and command injection, target the vulnerabilities of business applications, which generally have inadequate parameter checking.

However, if a business depends heavily on the logic of a business application, its input parameters must be checked.

Wivern uses the servlet filter[4] in a Web container in the preceding processing of a business application that extends to JSP and the Servlets, and inspects and processes requests relayed from the HTTP server to the Web container. The Servlet filter is a new function that has been added to the Servlet API2.3[5] interface filter for forwarding requests to and responses from a Web container.

Figure 4 shows the functional components of Wivern.

Wivern has the following features:

1) Implementation without changing existing applications

Because Wivern is implemented as a Java Servlet filter, it can be implemented without making design changes or modifying existing applications.

2) Increased efficiency of development work

Because the operation processing part and security processing part are independent, Wivern can be operated while these two parts are being developed. Moreover, the inspection processing

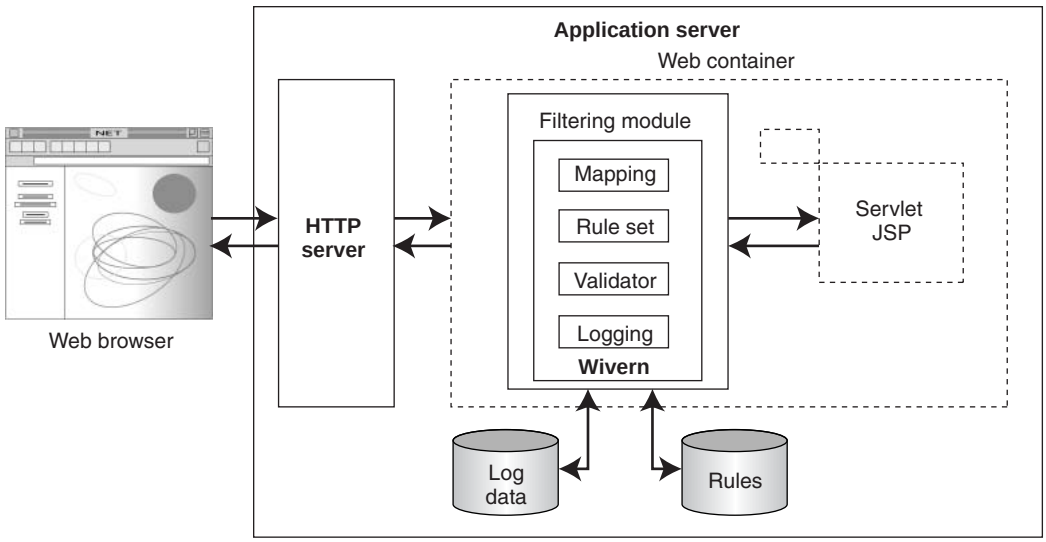

Figure 4
Function of Wivern.

of typical input values is done using six types of tester parts having a total of 20 test functions, and in most cases, the only work required for Wivern implementation is to describe a rule file and setting file. Moreover, the user can additionally create original inspection components as required.

3) Flexible error processing

Unlike the case for Web application firewalls, a standard method is used to interface Wivern with Web applications. Therefore, when abnormalities are detected by input inspection, processing can be done in a format standardized for business applications.

4) No need to periodically renew rule files

Because inspection is conducted using a positive approach (with the system allowing only safe content to pass), there is no need to renew rule files unless there are extensions and changes to an application.

5.3 Performance evaluation

The real-time filter and Wivern offer a filtering capability on the Web application input side.

Table 1 shows how much these filters affect performance in terms of the time needed to process a Web application request.

The table shows that the real-time filter and Wivern reduce performance by approximately 8% and 9%, respectively, which are almost the same reductions as the approximately 10% degradation in performance that is generally attributed to a Firewall.

Moreover, there is a large amount of Web application traffic on the output side and the traffic rate on the input side, which is used as a candidate for filtering, is expected to be no more than 10% of that on the output side. Therefore, the time overhead of these filters is low compared with the time required to process a Web application.

Since these products were made as an experiment to evaluate each function, a comprehensive, overall performance evaluation was not done.

6. Technical trend

To protect application servers, security equipment is being placed before the HTTP server to enable detection and protection against illegal access at the HTTP level (**Figure 5**).

It is difficult to perform fine control in conjunction with business applications protected by this equipment. However, the main aspects of the application server framework can be controlled at a level near the level of the business application.

Although security functions such as authentication are gradually being implemented in the J2EE framework, there is currently no overall security framework to protect against illegal access. However, there is a growing trend toward implementing such a framework on the framework side of application servers.

7. Conclusions

This paper described a framework for protecting against illegal access of an application server. We showed that this framework can protect an application server against the main types of vulnerability in Web applications, for example, session hijacks, cross-site scripting (XSS), and command injection. In the framework, a real-time filter supervises the stream of HTTP on an HTTP server to defend against input requests and Wivern supervises the display screen in a Web container.

Table 1
Performance degradation.

Component	Before (ms)	After (ms)	Performance degradation (%)	Measurement environment
Real-time filter	19.4	21	8	Pentium III 600 MHz Turbo Linux 6.0
Wivern	14.6	15.9	9	Pentium III 800 MHz Tomcat 4, Struts 1.1, Linux

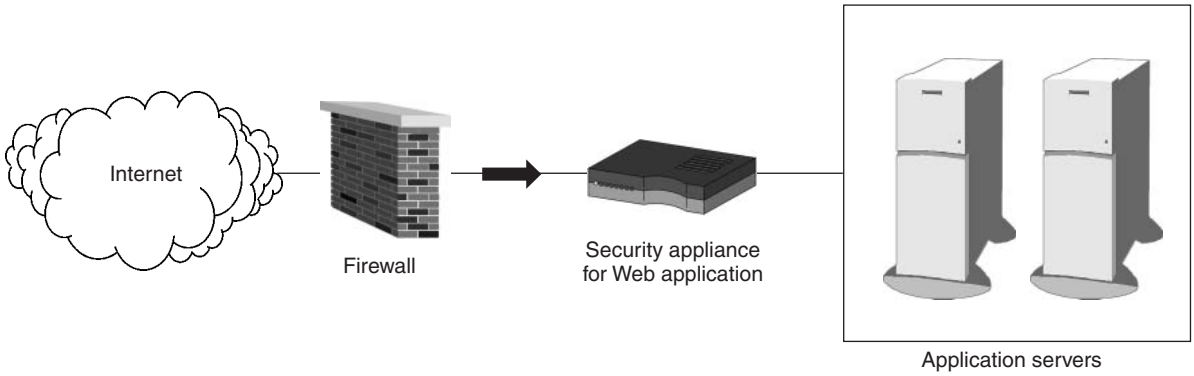

Figure 5
Web security appliance.

We are currently considering the functions of frameworks used to prevent illegal accesses for implementation in the Interstage Application Server. While working to achieve the remaining functions for making components more secure, we will adopt and maintain compatibility with the new development techniques for safe software that are being advanced by current research.

References

1) The Ten Most Critical Web Application Security Vulnerabilities. The Open Web Application Security Project (OWASP), January 13, 2003. *http://www.owasp.org/documentation/topten*
2) M. Mitomo et al.: A practical evaluation of the filtering tool for Web servers. (in Japanese), 15th Information Processing Society of Japan, CSEC study group report, December 21, 2001.
3) I. Morikawa et al.: Input Validation Filter for Java Servlet. 19th Annual Computer Security Applications Conference, December 8-12, 2003.
4) Core J2EE Patterns - Intercepting Filter, Core J2EE Pattern Catalog. *http://java.sun.com/blueprints/corej2eepatterns/Patterns/InterceptingFilter.html*
5) Servlet API 2.3, Interface Filter. *http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/Filter.html*

Satoru Torii received the B.S. degree in Information Sciences from Tokyo University of Science, Chiba, Japan in 1985. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1985, where he has been engaged in research and development of operating systems, intrusion management systems, and information security architecture. He is a member of the Information Processing Society of Japan (IPSJ).

Yoshiki Higashikado received the B.S. degree in Chemical Physics from Ritsumeikan University, Kyoto, Japan in 1990. He joined Fujitsu Ltd., Shinyokohama, Japan in 1990, where he has been engaged in development of networking systems. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 2003, where he has been engaged in research and development of networking systems, intrusion management systems, and information security architecture. He is a member of the Japan Society for Software Science and Technology (JSSST).

Takayoshi Kurita received the B.S. degree in Information Sciences from Ibaraki University, Ibaraki, Japan in 1986. He joined Fujitsu Ltd., Shinyokohama, Japan in 1986, where he has been engaged in development of Public-Key Infrastructure (PKI) and its applications.