AppSensor Developer Guide

From OWASP

Contents

- 1 AppSensor Developer Guide
  - 1.1 Developer Overview
  - 1.2 Obtaining AppSensor
  - 1.3 Extending AppSensor
    - 1.3.1 IntrusionStore
    - 1.3.2 ResponseAction
    - 1.3.3 ASUtilities
    - 1.3.4 TrendLogger
  - 1.4 Adding AppSensor to your project
    - 1.4.1 Dependencies
    - 1.4.2 With Maven
    - 1.4.3 Without Maven
  - 1.5 Using AppSensor in your project
    - 1.5.1 Intrusion Detection Code
    - 1.5.2 Intrusion Threshold Configuration

AppSensor Developer Guide

The AppSensor Project describes an application layer intrusion detection system. There is a Java implementation of this system whose basic usage can be found in the Getting Started guide. This document describes in more technical detail for developers how to use and extend AppSensor for a specific environment and application.

Developer Overview

AppSensor is an application layer intrusion detection system. The concept in implementation is roughly analogous to an intrusion detection (and prevention) system in the network security world. However, this concept can be applied inside of an application in a more specific way that (importantly) reduces false positives, which is an issue that often plagues network intrusion detection systems. This means that the core of the AppSensor system performs detection, monitoring, and (possibly) response depending on configuration settings.

AppSensor has been built to be quite extensible from the ground up. Most of the system can be appreciably modified to your needs by simply extending certain key interfaces, and modifying the appsensor configuration file appropriately. This extensible design makes it possible for various configurations to be applied depending upon the application. For instance, in a small application, you may choose to use a simple file-based model for storing intrusions that are detected, whereas for a larger application, you may have a relational database serving as your data store. See the **Extending AppSensor** section below for specifics on what can be extended.

A fuller description of the detection points (http://www.owasp.org/index.php /AppSensor_DetectionPoints) and responses (http://www.owasp.org/index.php /AppSensor_ResponseActions) are available - note that not all of these ideas are implemented here. The selection of detection points, where they are added, and how the application responds is application and organisation dependent.

Obtaining AppSensor

The appsensor.jar can be downloaded here - AppSensor-0.1.3.jar (http://repo2.maven.org/maven2 /org/owasp/appsensor/AppSensor/0.1.3/AppSensor-0.1.3.jar)

The source is available here (http://code.google.com/p/appsensor/source/browse/#svn/trunk /AppSensor%3Fstate%3Dclosed)

Extending AppSensor

Below you will find the individual interfaces you are likely to extend in order to modify AppSensor for your environment.

IntrusionStore

The IntrusionStore (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main /java/org/owasp/appsensor/intrusiondetection/IntrusionStore.java) interface represents, simply enough, the storage mechanism for any intrusions that occur in the system. The AppSensorIntrusionDetector (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp /appsensor/intrusiondetection/AppSensorIntrusionDetector.java) takes care of adding the intrusions to the intrusion store. The current DefaultIntrusionStore (http://code.google.com/p/appsensor/source /browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/reference /DefaultIntrusionStore.java) class stores all of the intrusions in a simple HashMap. The class is fairly small and simple, though, so it is trivial to understand it's inner workings and to use similar concepts to build an implementation that suits your environment.

```
The setting you'll need to modify in appsensor.properties to enable your own implementation is:

# This is the class that handles the intrusion store
AppSensor.intrusionStore=org.owasp.appsensor.intrusiondetection.reference.DefaultIntrusionStore
```

ResponseAction

The ResponseAction (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/ResponseAction.java) interface is used to simply respond to an intrusion once a threshold has been crossed. The decision for when to respond is handled by the AppSensorIntrusionDetector (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/AppSensorIntrusionDetector.java) , but the actual handling of the response is delegated to implementations of this interface. The only reason to not use the DefaultResponseAction (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/reference/DefaultResponseAction.java) would be if you have additional response actions you need to take, or if you need to modify the handling of one of the existing responses. Again, the DefaultResponseAction (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/reference/DefaultResponseAction.java) should give you a good starting point for creating your own implementation.

```
The setting you'll need to modify in appsensor.properties to enable your own implementation is:

# This is the class that handles the response actions
AppSensor.responseAction=org.owasp.appsensor.intrusiondetection.reference.DefaultResponseAction
```

DefaultResponseAction (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/intrusiondetection/reference/DefaultResponseAction.java) supports the following actions:

- "log" - logs the activity
- "logout" - logs the currently logged in user out (if one exists)
- "disable" - disables the account of the currently logged in user (if one exists)
- "disableComponent" - disables access to the location of the intrusion using the AppSensorServiceController
- "disableComponentForUser" - disables access to the location of the intrusion using the AppSensorServiceController for the currently logged in user(if one exists)
- "emailAdmin" - Email administrator to notify of what action has occurred
- "smsAdmin" - SMS administrator (via email to sms account) to notify of what action has occurred

```
Response actions can be configured for individual detection points using properties like: IntrusionDetect

# list of actions you want executed in the specified order as the threshold for this intrusion is met -
# ie. log the first time, logout the user the second time, etc.
IntrusionDetector.IE99.actions=log,logout,disable,disableComponent
```

ASUtilities

The ASUtilities (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/ASUtilities.java) interface handles a collection of concerns. It handles retrieving the current user of the application (ie. the user that made the current request and/or caused the current intrusion). In addition, it handles the retrieval of the logger as well as the current HTTP request. The DefaultASUtilities (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/reference/DefaultASUtilities.java) implementation simply delegates to the equivalent ESAPI method calls to retrieve the appropriate data. If you are not using ESAPI's logging and/or authentication and/or request binding utilities, you'll need to create your own implementation of this class. ***Note: This is the interface you'll need to implement if you are not using ESAPI.***

```
The setting you'll need to modify in appsensor.properties to enable your own implementation is:

# This is the class that handles the utility retriever
AppSensor.asUtilities=org.owasp.appsensor.reference.DefaultASUtilities
```

TrendLogger

The TrendLogger (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/trendmonitoring/TrendLogger.java) interface is in place to handle the logging of events in order to monitor trends. The techniques for doing this vary widely depending upon environment. You'll most likely not want to use the existing InMemoryTrendLogger (http://code.google.com/p/appsensor/source/browse/trunk/AppSensor/src/main/java/org/owasp/appsensor/trendmonitoring/reference/InMemoryTrendLogger.java) as it will scale very poorly. It is simply in place as a starting point for other implementations.

```
The setting you'll need to modify in appsensor.properties to enable your own implementation is:

# This is the class that handles the trend logging
AppSensor.trendLogger=org.owasp.appsensor.trendmonitoring.reference.InMemoryTrendLogger
```

Adding AppSensor to your project

Dependencies

AppSensor has the following dependencies:

- OWASP ESAPI Java library
- JavaMail libraries (activation and mail jar files)
- Servlet/JSP libraries
- Logging API library (log4j by default)

With Maven

If you use maven as the build system for your application, then adding AppSensor to your project is very simple and requires 4 basic steps.

1. Add the following configuration into the **dependencies** section of your POM:

   ```
   ```

   <dependency>
   <groupId>org.owasp.appsensor</groupId>
   <artifactId>AppSensor</artifactId>
   <version>PUT_YOUR_VERSION_HERE</version>
   </dependency>
2. Add the **.esapi** folder to your project in a location that ESAPI can find it. The easiest way to do this is to add the folder under the root of your **src** folder. Additionally, you will need to place 3 files in the .esapi folder:
   - ESAPI.properties
   - validation.properties
   - appsensor.properties
3. Modify the following line in the ESAPI.properties file:

   ```
   ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
   ```

   The line above should be changed to:

   ```
   ESAPI.IntrusionDetector=org.owasp.appsensor.intrusiondetection.AppSensorIntrusionDetector
   ```

4. Customize the configuration files listed above (those in the .esapi folder) to suit your own project.
5. **TODO:** Add example appsensor configuration or examples

At this point you should have the project configured to work properly.

Without Maven

If you use some mechanism other than maven as the build system for your application, then adding AppSensor requires downloading and adding all required libraries to your project manually in addition to the other basic steps, and this process is outlined below:

1. Download the libraries described as dependencies above, and add them to your project (likely in the **WEB-INF/lib** folder of your application).
2. Add the **.esapi** folder to your project in a location that ESAPI can find it. The easiest way to do this is to add the folder under the root of your **src** folder. Additionally, you will need to place 3 files in the .esapi folder:
   - ESAPI.properties
   - validation.properties
   - appsensor.properties
3. Modify the following line in the ESAPI.properties file:

   ```
   ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
   ```

The line above should be changed to

```
ESAPI.IntrusionDetector=org.owasp.appsensor.intrusiondetection.AppSensorIntrusionDetector
```

4. Customize the configuration files listed above (those in the .esapi folder) to suit your own project.

At this point you should have the project configured to work properly.

Using AppSensor in your project

Once your project is properly configured to use AppSensor, using AppSensor is very simple. It takes only 2 basic steps: **Intrusion Detection Code** and **Intrusion Threshold Configuration**.

Intrusion Detection Code

Here are a couple of very simple examples.

The following example involves creating an AppSensorException by hand in your application:

```
//This example snippet might be placed on a jsp that handles HTTP 404 errors.
//When the page is accessed, this code notifies AppSensor that an invalid page request was made.
//Notice that the exception is created, not thrown

new AppSensorException("ACE3", "Invalid request", "Attacker is requesting a non-existent (404) page (" +
```

The following example relies on the AttackDetectorUtils class to create the exception. This class contains various methods that handle common detection points.

```
//This example snippet might be placed in request handling code that expects a form POST to occur (not a
//This code notifies AppSensor that an type of HTTP request was made.

AttackDetectorUtils.verifyValidRequestMethod(request, AttackDetectorUtils.POST);
```

Intrusion Threshold Configuration

Here the salient points to note are that the configuration can vary depending on what response actions are desired as well as what threshold is required. Additionally, you have control over these settings and can tune them to produce the desired effect. These configurations can be set in the appsensor.properties file in your .esapi folder. *Note: While these threshold configurations can be set in the ESAPI.properties in the .esapi folder, it is not recommended for AppSensor so that the configuration can be kept together in one place. Also note some thresholds may already be configured for subclasses of ESAPI's EnterpriseSecurityException in the ESAPI.properties file - please be aware of that when setting up the configuration for your application.*

Below are two sample threshold configurations for two separate detection points.

```
# http://www.owasp.org/index.php/AppSensor_DetectionPoints#ACE2:_Modifying_Parameters_Within_A_POST_For_D
# number of intrusions in a specified segment of time that constitutes the upper threshold - once crosse
IntrusionDetector.ACE2.count=3
# segment of time (in seconds)
IntrusionDetector.ACE2.interval=3
# list of actions you want executed in the specified order as the threshold for this intrusion is met - i
IntrusionDetector.ACE2.actions=log,logout,disable,disableComponent
# some integer - duration of time to disable
IntrusionDetector.ACE2.disableComponent.duration=30
# some measure of time, currently supported are s,m,h,d (second, minute, hour, day)
IntrusionDetector.ACE2.disableComponent.timeScale=m
# some integer - duration of time to disable
IntrusionDetector.ACE2.disableComponentForUser.duration=30
# some measure of time, currently supported are s,m,h,d (second, minute, hour, day)
IntrusionDetector.ACE2.disableComponentForUser.timeScale=m
```

```
# http://www.owasp.org/index.php/AppSensor_DetectionPoints#RE3:_GET_When_Expecting_POST
# number of intrusions in a specified segment of time that constitutes the upper threshold - once crosse
IntrusionDetector.RE3.count=3
# segment of time (in seconds)
IntrusionDetector.RE3.interval=300
# list of actions you want executed in the specified order as the threshold for this intrusion is met - i
IntrusionDetector.RE3.actions=log,logout,disable
```

This document shows that AppSensor is very simple to configure as well as to use. Once setup, you simply add detection points to your code and add and/or modify the appropriate configuration information in the ESAPI.properties and appsensor.properties files in order to let AppSensor know your appropriate thresholds for each detection point. That's it!

Retrieved from "http://www.owasp.org/index.php/AppSensor_Developer_Guide"

- This page was last modified on 19 October 2010, at 05:16.
-