Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Code Inspection

Student: Federico Gatti [Matricola: 852377]
Student: Luca Fochetta [Matricola: 792935]

AA 2015–2016

Contents

Chapter 1

Assigned Class

The class assigned us is the *CSIV2TaggedComponentInfo* and the methods are the following:

- **Name**: createSASContextSec(EjbIORConfigurationDescriptor iorDesc)
  - **Location**: appserver/security/ejb.security/src/main/java/com/sun/enterprise/iiop/security/CSIV2TaggedComponentInfo.java
  - **Start Line**: 512
  - **End Line**: 566
  - **Number of Line**: 54
- **Name**: getTargetSupports(EjbIORConfigurationDescriptor iorDesc)
  - **Location**: appserver/security/ejb.security/src/main/java/com/sun/enterprise/iiop/security/CSIV2TaggedComponentInfo.java
  - **Start Line**: 571
  - **End Line**: 599
  - **Number of Line**: 29
- **Name**: getTargetRequires(EjbIORConfigurationDescriptor iorDesc)
  - **Location**: appserver/security/ejb.security/src/main/java/com/sun/enterprise/iiop/security/CSIV2TaggedComponentInfo.java
  - **Start Line**: 604
  - **End Line**: 632
  - **Number of Lines**: 29
- **Name**: createSSLInfo(int sslport, EjbIORConfigurationDescriptor iorDesc, boolean sslRequired)

  - **Location**: appserver/security/ejb.security/src/main/java/com/sun/enterprise/iiop/security/CSIV2TaggedComponentInfo.java
  - **Start Line**: 706
  - **End Line**: 734
  - **Number of Lines**: 29

- **Name**: getSecurityMechanisms(IOR ior)

  - **Location**: appserver/security/ejb.security/src/main/java/com/sun/enterprise/iiop/security/CSIV2TaggedComponentInfo.java
  - **Start Line**: 792
  - **End Line**: 827
  - **Number of Lines**: 36

Chapter 2

Functional Role

In this chapter is described the main role of the class *CSIV2TaggedComponentInfo* and the tasks performed by the method assigned us.

2.1 Main Role

The *CSIV2TaggedComponentInfo* manages the CSIV2 tagged component information in the IORs.

In distributed computing, CSIv2 (Common Secure Interoperability Protocol Version 2) is a protocol implementing security features for inter-ORB communication. It intends, in part, to address limitations of SSLIOP.

An **object request broker (ORB)** is a middleware which allows program calls to be made from one computer to another via a computer network, providing location transparency through remote procedure calls. ORBs promote interoperability of distributed object systems, enabling such systems to be built by piecing together objects from different vendors, while different parts communicate with each other via the ORB.

2.2 Methods Role

2.2.1 createSASContextSec

Create the SAS layer context within a compound mechanism definition.

2.2.2 getTargetSupports

Get the value of *target_supports* for the transport layer.

2.2.3 getTargetRequires

Get the value of *target_requires* for the transport layer.

2.2.4 createSSLInfo

Create the *SSL tagged component* within a compound mechanism definition.

2.2.5 getSecurityMechanisms

Get the Compound security mechanism list from the given IOR. Return the array of *compound security mechanisms*.

Chapter 3

List of issues

In this chapter are listed all the issues found by applying the checklist

3.1 Naming Conventions

We don't found any problems with this issue. The only doubt coming out with the line:

```
private static final Logger _logger ;
```

because *_logger* is a constant so the name attribute must be capitalized, but the *logger* is the only exception, so it doesn't follow this rule.

3.2 Indention

3.2.1 Use of the Tab

9. No tabs are used to indent
There are several problems with the use of the tab used to indent the code.

3.2.1.1 tab in createSASContextSec

The lines where is used the tab to indent are:

1. 513
2. 514
3. 517
4. 529
5. 530
6. 537

7. 541
8. 543
9. 545
10. From 553 to 557
11. 563

3.2.1.2 tab in getTargetSupports

The lines where is used the tab to indent are:

1. 573
2. From 577 to 598

3.2.1.3 tab in getTargetRequires

The lines where is used the tab to indent are:

1. 606
2. from 610 to 614
3. from 616 to 619
4. from 621 to 624
5. from 626 to 629
6. 631

3.2.1.4 tab in createSSLInfo

1. 707
2. 708
3. from 710 to 714
4. 716
5. 719
6. 729
7. 730
8. 732
9. 733

3.2.1.5 tab in getSecurityMechanisms

1. from 793 to 797
2. 799
3. from 803 to 805
4. from 807 to 810
5. from 812 to 814
6. 816
7. 817
8. from 819 to 824
9. 826

3.3 Braces

There aren't problems of this type

3.4 File Organization

3.4.1 line that exceed 80 characters

13. Where practical, line length does not exceed 80 characters.

3.4.1.1 line that exceed 80 characters in createSASContextSec

The line 537 exceed the 80 characters

```
&&(callerPropagation.equalsIgnoreCase(EjbIORConfigurationDescriptor.NONE))){
```

3.4.1.2 line that exceed 80 characters in getTargetSupports

The lines that exceed the 80 characters are:

1. 584

   ```
   if(!confidentiality.equalsIgnoreCase(EjbIORConfigurationDescriptor.NONE)){
   ```

2. 589

   ```
   if(!establishTrustInTarget.equalsIgnoreCase(EjbIORConfigurationDescriptor.NONE)){
   ```

3. 594

```
if(!establishTrustInClient.equalsIgnoreCase(EjbIORConfigurationDescriptor.NONE)){
```

3.4.1.3 line that exceed 80 characters in getTargetRequires

1. 617

```
if(confidentiality.equalsIgnoreCase(EjbIORConfigurationDescriptor.REQUIRED)){
```

2. 622

```
if(establishTrustInTarget.equalsIgnoreCase(EjbIORConfigurationDescriptor.REQUIRED)){
```

3. 627

```
if(establishTrustInClient.equalsIgnoreCase(EjbIORConfigurationDescriptor.REQUIRED)){
```

3.4.1.4 line that exceed 80 characters in createSSLInfo

1. 718

```
_logger.log(Level.FINE, "IIOP: Creating Transport Mechanism for sslport "
```

2. 733

```
return createTlsSecTransComponent( targetSupports, targetRequires, listTa );
```

3.4.1.5 line that exceed 80 characters in getSecurityMechanisms

1. 801

```
String msg = "IIOP:TAG_CSI_SEC_MECH_LIST tagged component not found";
```

2. 821

```
CDRInputObject in = (CDRInputObject) new EncapsInputStream(orb, b,
    b.length);
```

3.5 Wrapping Lines

3.5.1 Line break

15. Line break occurs after a comma or an operator.

3.5.1.1 Line break in createSASContextSec

In the lines 536-537 is present a line break before the && operator

```
if ((callerPropagation != null)
&&(callerPropagation.equalsIgnoreCase(EjbIORConfigurationDescriptor.NONE))){
```

3.5.1.2 Line break in createSSLInfo

In the line 718-719 is present a line break before the + operator

```
_logger.log(Level.FINE, "IIOP: Creating Transport Mechanism for sslport "
+ ssl_port );
```

3.6 Comments

There aren't problems of this type

3.7 Java Source Files

3.7.1 JavaDoc

23. Check that the javadoc is complete

3.7.1.1 Javadoc for createSASContextSec

SAS_ContextSec and ServiceConfiguration javadoc aren't present.

3.7.1.2 Javadoc for createSSLInfo

createSSLInfo and createTlsSecTransComponent aren't present.

3.8 Package and Import Statements

There aren't problems of this type

3.9 Class and Interface Declarations

A possible problem can be the repetition of a piece of code in the method getTarget-Support and getTargetRquires, but the repetition is very limited and to avoid it is needed a pattern that will be make hard to understand the code. So the repetition is justify for us.

3.10 Initialization and Declarations

3.10.1 Variables declaration

33. Declarations appear at the beginning of blocks

3.10.1.1 Declarations in createSASContextSec

In the line 530 there is the declaration and the initialization of the variable *supported_ identity_ token_ type* but it isn't at the beginning of a blocks.

```
int target_supports = 0;
int target_requires = 0;
ServiceConfiguration[] priv = new ServiceConfiguration[0];
String callerPropagation = null;
byte[][] mechanisms = {};
if(_logger.isLoggable(Level.FINE)){
   _logger.log(Level.FINE, "IIOP: Creating SAS_Context");
}
// this shall be non-zero if target_supports is non-zero
int supported_identity_token_type = 0;
```

3.10.1.2 Declarations in getTargetSupports

In the lines 577-578-583-588-593 appear declarations and initialization of the variable:

1. *supports*
2. *integrity*
3. *confidentiality*
4. *establishTrustInTarget*
5. *establishTrustInClient*

3.10.1.3 Declaration in getTargetRequires

In the lines 610-611-616-621-626 appear declarations and initialization of the variables:

1. *requires*
2. *integrity*
3. *confidentiality*
4. *estabishTrustInTarget*
5. *estabilishTrustInClient*

3.10.1.4 Declaration in createSSLInfo

In the line 732 appear declaration and initialization of the variable *listTa* but it isn't at the beggining of a block.

```
if ( (targetSupports | targetRequires) == 0 || ssl_port == -1) {
   return NULL_TAGGED_COMPONENT ;
}

TransportAddress[] listTa = generateTransportAddresses( ssl_port ) ;
```

3.10.1.5 declaration in getSecurityMechanisms

In the line 807-813-819-820-821-823-824 appear declarations and initialization of the variables:

1. *tcomp*
2. *msg*
3. *comp*
4. *b*
5. *in*
6. *l*
7. *list*

3.11 Methods Calls

3.11.1 Parameters in the correct order

34. Check that parameters are presented in the correct order

3.11.1.1 Parameters in the correct order in createSASContextSec

Considering that isn't present the javadoc for some of the object used in this class we can't say anything about the correctness of the methods call. Obviously Java compiler checks that all the parameters are of the correct type, but we can't know if two or more parameters are swapped in the call.

3.11.1.2 Paramenters in the correct order in createSSLInfo

Considering that this method isn't in the javadoc like the method *createTlsSecTransComponent* which is called inside, we can't say anything about the correctness of the methods call. For *createTlsSecTransComponent* we can say that the attributes' name used inside

the method are the same of the one used in *createSSLInfo* and they are inserted in the method call in the same order, despite this we can't say if the attributes' name in both the method mean the same thing. Obviously Java compiler checks that all the parameters are of the correct type, but we can't know if two or more parameters are swapped in the call.

3.12 Arrays

There aren't problem of this type

3.13 Object Comparison

There aren't problem of this type

3.14 Output Format

There aren't problem of this type

3.15 Computation, Comparisons and Assignments

3.15.1 Brutish programming

44. Check that the implementation avoids «brutish programming»

3.15.2 Brutish programming in createSASContextSec

In the line 549-550-551 there is this code:

```
for(int i = 0; i < upm.length; i++) {
   mechanisms[0][i] = upm[i];
}
```

that is a manual array copy, as suggest java is better use the method

```
System.arraycopy(upm, 0, mechanisms[0], 0, upm.length);
```

3.16 Exceptions

There aren't problems of this type

3.17 Flow of Control

There aren't problems of this type

3.18 Files

There aren't problems of this type

Chapter 4

Other Problem

In this chapter are described some problems don't list in the list of issues:

1. The line 545 must be shifted to line 555 to improve the comprehension of the code
2. Mixed indentation
   (a) Line 523
   (b) Line 716
3. Use of the *if-else* starting from line 536 until the end to improve the comprehension of the code
4. The code starting from line 538 to 542 is repeated in the lines 560-565. It's better don't repeat the code and modify the value of the parameters using the *if-else* structure
5. Spaces at the end of the line
   (a) Line 536
   (b) Line 707
   (c) Line 715
   (d) Line 722 and 723
   (e) Line 813