Fortify Security Report

Jun 17, 2015

dkwakkel

Executive Summary

Issues Overview

On Jun 2, 2015, a source code review was performed over the poi code base. 3,213 files, 178,513 LOC (Executable) were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 2342 reviewed findings were uncovered during the analysis.

Issues by Fortify Priority Order

Refined by: category:"system information leak\: external"

Low	36

Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level. The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

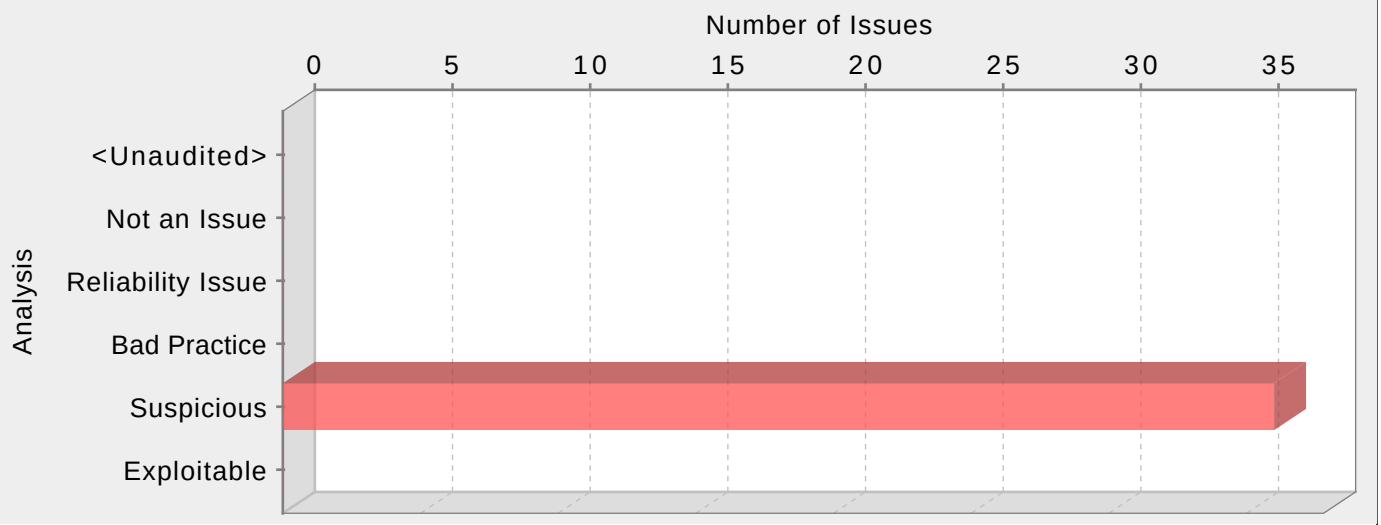
Results Outline

Overall number of results

The scan found 2342 issues.

Vulnerability Examples by Category

Category: System Information Leak: External (36 Issues)



Abstract:

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation:

An external information leak occurs when system data or debugging information leaves the program to a remote machine via a socket or network connection. External leaks can help an attacker by revealing specific data about operating systems, full pathnames, the existence of usernames, or locations of configuration files, and are more serious than internal information leaks which are more difficult for an attacker to access.

Example 1: The following code leaks Exception information in the HTTP response:

```
protected void doPost (HttpServletRequest req, HttpServletResponse res) throws IOException {
...
PrintWriter out = res.getWriter();
try {
...
} catch (Exception e) {
out.println(e.getMessage());
}
}
```

This information can be exposed to a remote user. In some cases the error message tells the attacker precisely what sort of an attack the system is vulnerable to. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In the example above, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

In the mobile world, information leaks are also a concern. The essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which is why application authors need to be careful about what information they include in messages addressed to other applications running on the device.

Example 2: The code below broadcasts the stack trace of a caught exception to all the registered Android receivers.

```
...
try {
...
} catch (Exception e) {
String exception = Log.getStackTraceString(e);
```

```
Intent i = new Intent();
i.setAction("SEND_EXCEPTION");
i.putExtra("exception", exception);
view.getContext().sendBroadcast(i);
}
...
```

Here is another scenario specific to the mobile world. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices to close proximity or simply having them touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, since NFC alone does not ensure secure communication.

Example 3: The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within the range.

```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
TelephonyManager tm = (TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
String VERSION = tm.getDeviceSoftwareVersion();
...
NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (nfcAdapter == null)
return;
String text = TAG + DATA_SPLITTER + VERSION;
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
MIME_TYPE.getBytes(), new byte[0], text.getBytes());
NdefRecord[] records = { record };
NdefMessage msg = new NdefMessage(records);
nfcAdapter.setNdefPushMessage(msg, this);
...
```

NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper.

Recommendations:

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Be careful, debugging traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example).

Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. Due to this, it's advised to always keep information instead of sending it to a resource directly outside the program.

Example 4: The code below broadcasts the stack trace of a caught exception within your app only, so that it cannot be leaked to other apps on the system. There is also the added bonus that this is more efficient than globally broadcasting through the system.

```
...
try {
...
} catch (Exception e) {
String exception = Log.getStackTraceString(e);
Intent i = new Intent();
i.setAction("SEND_EXCEPTION");
i.putExtra("exception", exception);
LocalBroadcastManager.getInstance(view.getContext()).sendBroadcast(i);
}
...
```

If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Either do not include system data in the messages pushed to other devices in range, or encrypt the payload of the message, or establish secure communication channel at a higher layer.

Tips:

1. Do not rely on wrapper scripts, corporate IT policy, or quick-thinking system administrators to prevent system information leaks. Write software that is secure on its own.

2. This category of vulnerability does not apply to all types of programs. For example, if your application executes on a client machine where system information is already available to an attacker, or if you print system information only to a trusted log file, you can use AuditGuide to filter out this category.

ChunkedCipherOutputStream.java, line 127 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function writeChunk() in ChunkedCipherOutputStream.java might reveal system data or debugging information by calling write() on line 127. The information revealed by write() could help an adversary form a plan of attack.		

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: ChunkedCipherOutputStream.java:127 java.io.OutputStream.write()

```
125
126             int ciLen = _cipher.doFinal(_chunk, 0, posInChunk, _chunk);
127             out.write(_chunk, 0, ciLen);
128         }
129
```

Analysis: Suspicious

NDocumentOutputStream.java, line 123 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 123. The information revealed by write() could help an adversary form a plan of attack.		

Source: NPOIFSDocument.java:225 java.lang.Throwable.getMessage()

```
223             }
224         } catch (IOException e) {
225             result = e.getMessage();
226         }
227         results[0] = result;
```

Sink: NDocumentOutputStream.java:123 java.io.OutputStream.write()

```
121
122             if (_buffer != null) {
123                 _buffer.write(b);
124                 checkBufferSize();
125             } else {
```

Analysis: Suspicious

NDocumentOutputStream.java, line 126 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 126. The information revealed by write() could help an adversary form a plan of attack.		

Source: HSLFSlideShow.java:358 Read ie()

```
356         currentUser = new CurrentUserAtom(directory);
```

```
357                 } catch(IOException ie) {
358                  logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
359                  currentUser = new CurrentUserAtom();
360                 }
```

Sink: NDocumentOutputStream.java:126 org.apache.poi.poifs.filesystem.NDocumentOutputStream.write()

```
124                     checkBufferSize();
125                 } else {
126                     write(b, 0, b.length);
127                 }
128              }
```

Analysis: Suspicious

LittleEndianOutputStream.java, line 84 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in LittleEndianOutputStream.java might reveal system data or debugging information by calling write() on line 84. The information revealed by write() could help an adversary form a plan of attack.

Source: POIFSDocument.java:367 java.lang.Throwable.getMessage()

```
365                 }
366                 } catch (IOException e) {
367                  result = e.getMessage();
368                 }
369                 results[0] = result;
```

Sink: LittleEndianOutputStream.java:84 java.io.FilterOutputStream.write()

```
82                  // suppress IOException for interface method
83                  try {
84                   super.write(b);
85                  } catch (IOException e) {
86                   throw new RuntimeException(e);
```

Analysis: Suspicious

MemoryPackagePartOutputStream.java, line 93 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in MemoryPackagePartOutputStream.java might reveal system data or debugging information by calling write() on line 93. The information revealed by write() could help an adversary form a plan of attack.

Source: HSLFSlideShow.java:358 Read ie()

```
356                   currentUser = new CurrentUserAtom(directory);
357                 } catch(IOException ie) {
358                   logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
359                   currentUser = new CurrentUserAtom();
360                 }
```

Sink: MemoryPackagePartOutputStream.java:93 java.io.OutputStream.write()

```
91                  @Override
92                  public void write(byte[] b) throws IOException {
93                   _buff.write(b);
94                  }
95                 }
```

Analysis: Suspicious

ChunkedCipherOutputStream.java, line 81 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract:	The function write() in ChunkedCipherOutputStream.java might reveal system data or debugging information by calling write() on line 81. The information revealed by write() could help an adversary form a plan of attack.
Source:	POIFSDocument.java:367 java.lang.Throwable.getMessage()

```
365                 }
366               } catch (IOException e) {
367                 result = e.getMessage();
368               }
369               results[0] = result;
```

Sink:	ChunkedCipherOutputStream.java:81 org.apache.poi.poifs.crypt.ChunkedCipherOutputStream.write()

```
79
80                  public void write(byte[] b) throws IOException {
81                      write(b, 0, b.length);
82                  }
```

Analysis:	Suspicious

HWPFOutputStream.java, line 48 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in HWPFOutputStream.java might reveal system data or debugging information by calling write() on line 48. The information revealed by write() could help an adversary form a plan of attack.		
Source:	EscherBitmapBlip.java:129 Read e()		

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink:	HWPFOutputStream.java:48 java.io.ByteArrayOutputStream.write()

```
46              public synchronized void write(byte[] buf, int off, int len)
47              {
48                super.write(buf, off, len);
49                _offset += len;
50              }
```

Analysis:	Suspicious

ChunkedCipherOutputStream.java, line 81 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in ChunkedCipherOutputStream.java might reveal system data or debugging information by calling write() on line 81. The information revealed by write() could help an adversary form a plan of attack.		
Source:	HSLFSlideShow.java:358 Read ie()		

```
356               currentUser = new CurrentUserAtom(directory);
357             } catch(IOException ie) {
358               logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
359               currentUser = new CurrentUserAtom();
360             }
```

Sink:	ChunkedCipherOutputStream.java:81 org.apache.poi.poifs.crypt.ChunkedCipherOutputStream.write()

```
79
80                  public void write(byte[] b) throws IOException {
81                      write(b, 0, b.length);
82                  }
```

Analysis:	Suspicious

LittleEndianOutputStream.java, line 84 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in LittleEndianOutputStream.java might reveal system data or debugging information by calling write() on line 84. The information revealed by write() could help an adversary form a plan of attack.		

Source: NPOIFSDocument.java:225 java.lang.Throwable.getMessage()

```
223                 }
224               } catch (IOException e) {
225                  result = e.getMessage();
226               }
227               results[0] = result;
```

Sink: LittleEndianOutputStream.java:84 java.io.FilterOutputStream.write()

```
82    // suppress IOException for interface method
83    try {
84     super.write(b);
85    } catch (IOException e) {
86     throw new RuntimeException(e);
```

Analysis: Suspicious

NDocumentOutputStream.java, line 123 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 123. The information revealed by write() could help an adversary form a plan of attack.		

Source: HSLFSlideShow.java:358 Read ie()

```
356     currentUser = new CurrentUserAtom(directory);
357    } catch(IOException ie) {
358     logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
359     currentUser = new CurrentUserAtom();
360    }
```

Sink: NDocumentOutputStream.java:123 java.io.OutputStream.write()

```
121
122             if (_buffer != null) {
123                 _buffer.write(b);
124                 checkBufferSize();
125             } else {
```

Analysis: Suspicious

MemoryPackagePartOutputStream.java, line 93 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in MemoryPackagePartOutputStream.java might reveal system data or debugging information by calling write() on line 93. The information revealed by write() could help an adversary form a plan of attack.		

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: MemoryPackagePartOutputStream.java:93 java.io.OutputStream.write()

```
91     @Override
92     public void write(byte[] b) throws IOException {
93      _buff.write(b);
94     }
95    }
```

Analysis: Suspicious

LittleEndianOutputStream.java, line 84 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in LittleEndianOutputStream.java might reveal system data or debugging information by calling write() on line 84. The information revealed by write() could help an adversary form a plan of attack.

Source: HSLFSlideShow.java:358 Read ie()

```
356                 currentUser = new CurrentUserAtom(directory);
357               } catch(IOException ie) {
358                 logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
359                 currentUser = new CurrentUserAtom();
360               }
```

Sink: LittleEndianOutputStream.java:84 java.io.FilterOutputStream.write()

```
82              // suppress IOException for interface method
83              try {
84                super.write(b);
85              } catch (IOException e) {
86                throw new RuntimeException(e);
```

Analysis: Suspicious

DocumentOutputStream.java, line 105 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in DocumentOutputStream.java might reveal system data or debugging information by calling write() on line 105. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: DocumentOutputStream.java:105 java.io.OutputStream.write()

```
103             {
104                 limitCheck(len);
105                 _stream.write(b, off, len);
106             }
```

Analysis: Suspicious

MemoryPackagePartOutputStream.java, line 88 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in MemoryPackagePartOutputStream.java might reveal system data or debugging information by calling write() on line 88. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: MemoryPackagePartOutputStream.java:88 java.io.ByteArrayOutputStream.write()

```
86              @Override
87              public void write(byte[] b, int off, int len) {
88                _buff.write(b, off, len);
89              }
```

Analysis:	Suspicious

ChunkedCipherOutputStream.java, line 81 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in ChunkedCipherOutputStream.java might reveal system data or debugging information by calling write() on line 81. The information revealed by write() could help an adversary form a plan of attack.		
Source:	EscherBitmapBlip.java:129 Read e()		

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: ChunkedCipherOutputStream.java:81 org.apache.poi.poifs.crypt.ChunkedCipherOutputStream.write()

```
79
80      public void write(byte[] b) throws IOException {
81          write(b, 0, b.length);
82      }
```

Analysis:	Suspicious

NDocumentOutputStream.java, line 126 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 126. The information revealed by write() could help an adversary form a plan of attack.		
Source:	NPOIFSDocument.java:225 java.lang.Throwable.getMessage()		

```
223              }
224            } catch (IOException e) {
225               result = e.getMessage();
226            }
227            results[0] = result;
```

Sink: NDocumentOutputStream.java:126 org.apache.poi.poifs.filesystem.NDocumentOutputStream.write()

```
124                 checkBufferSize();
125             } else {
126                 write(b, 0, b.length);
127             }
128         }
```

Analysis:	Suspicious

DocumentOutputStream.java, line 78 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in DocumentOutputStream.java might reveal system data or debugging information by calling write() on line 78. The information revealed by write() could help an adversary form a plan of attack.		
Source:	EscherBitmapBlip.java:129 Read e()		

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: DocumentOutputStream.java:78 org.apache.poi.poifs.filesystem.DocumentOutputStream.write()

```
76              throws IOException
```

```
77                  {
78                      write(b, 0, b.length);
79                  }
```

Analysis: Suspicious

DocumentOutputStream.java, line 78 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in DocumentOutputStream.java might reveal system data or debugging information by calling write() on line 78. The information revealed by write() could help an adversary form a plan of attack.

Source: POIFSDocument.java:367 java.lang.Throwable.getMessage()

```
365                 }
366               } catch (IOException e) {
367                 result = e.getMessage();
368               }
369               results[0] = result;
```

Sink: DocumentOutputStream.java:78 org.apache.poi.poifs.filesystem.DocumentOutputStream.write()

```
76                      throws IOException
77                  {
78                      write(b, 0, b.length);
79                  }
```

Analysis: Suspicious

NDocumentOutputStream.java, line 134 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 134. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: NDocumentOutputStream.java:134 java.io.ByteArrayOutputStream.write()

```
132
133                 if (_buffer != null) {
134                     _buffer.write(b, off, len);
135                     checkBufferSize();
136                 } else {
```

Analysis: Suspicious

LittleEndianOutputStream.java, line 84 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in LittleEndianOutputStream.java might reveal system data or debugging information by calling write() on line 84. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: LittleEndianOutputStream.java:84 java.io.FilterOutputStream.write()

```
82                  // suppress IOException for interface method
83                  try {
84                   super.write(b);
85                  } catch (IOException e) {
86                   throw new RuntimeException(e);
```

Analysis: Suspicious

CryptoAPIEncryptor.java, line 248 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in CryptoAPIEncryptor.java might reveal system data or debugging information by calling write() on line 248. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: CryptoAPIEncryptor.java:248 java.io.ByteArrayOutputStream.write()

```
246                 try {
247                     cipher.update(b, off, len, b, off);
248                     super.write(b, off, len);
249                 } catch (Exception e) {
250                     throw new EncryptedDocumentException(e);
```

Analysis: Suspicious

NDocumentOutputStream.java, line 123 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 123. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: NDocumentOutputStream.java:123 java.io.OutputStream.write()

```
121
122             if (_buffer != null) {
123                 _buffer.write(b);
124                 checkBufferSize();
125             } else {
```

Analysis: Suspicious

StreamHelper.java, line 69 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in StreamHelper.java might reveal system data or debugging information by calling write() on line 69. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
```

```
131                 StringBuilder builder = new StringBuilder();
```

Sink: StreamHelper.java:69 java.io.OutputStream.write()

```
67                  public void write(byte b[], int off, int len)
68                          throws IOException {
69                      out.write(b, off, len);
70                  }
```

Analysis: Suspicious

StandardEncryptor.java, line 151 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in StandardEncryptor.java might reveal system data or debugging information by calling write() on line 151. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: StandardEncryptor.java:151 java.io.OutputStream.write()

```
149                 @Override
150                 public void write(byte[] b, int off, int len) throws IOException {
151                     out.write(b, off, len);
152                     countBytes += len;
153                 }
```

Analysis: Suspicious

LittleEndianOutputStream.java, line 93 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in LittleEndianOutputStream.java might reveal system data or debugging information by calling write() on line 93. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: LittleEndianOutputStream.java:93 java.io.FilterOutputStream.write()

```
91                // suppress IOException for interface method
92                try {
93                 super.write(b, off, len);
94                } catch (IOException e) {
95                 throw new RuntimeException(e);
```

Analysis: Suspicious

DocumentOutputStream.java, line 78 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		

Abstract: The function write() in DocumentOutputStream.java might reveal system data or debugging information by calling write() on line 78. The information revealed by write() could help an adversary form a plan of attack.

Source: HSLFSlideShow.java:358 Read ie()

```
356               currentUser = new CurrentUserAtom(directory);
357             } catch(IOException ie) {
358               logger.log(POILogger.ERROR, "Error finding Current User Atom:\n" + ie);
```

```
359                 currentUser = new CurrentUserAtom();
360               }
```

Sink: DocumentOutputStream.java:78 org.apache.poi.poifs.filesystem.DocumentOutputStream.write()

```
76                  throws IOException
77              {
78                  write(b, 0, b.length);
79              }
```

Analysis: Suspicious

DocumentOutputStream.java, line 78 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in DocumentOutputStream.java might reveal system data or debugging information by calling write() on line 78. The information revealed by write() could help an adversary form a plan of attack.

Source: NPOIFSDocument.java:225 java.lang.Throwable.getMessage()

```
223                 }
224             } catch (IOException e) {
225                 result = e.getMessage();
226             }
227             results[0] = result;
```

Sink: DocumentOutputStream.java:78 org.apache.poi.poifs.filesystem.DocumentOutputStream.write()

```
76                  throws IOException
77              {
78                  write(b, 0, b.length);
79              }
```

Analysis: Suspicious

NDocumentOutputStream.java, line 141 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 141. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
128             {
129                 extraData = e.toString();
130             }
131             StringBuilder builder = new StringBuilder();
```

Sink: NDocumentOutputStream.java:141 java.io.OutputStream.write()

```
139                 _stream_output = _stream.getOutputStream();
140             }
141             _stream_output.write(b, off, len);
142             _document_size += len;
143         }
```

Analysis: Suspicious

NDocumentOutputStream.java, line 126 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 126. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127             catch ( Exception e )
```

```
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: NDocumentOutputStream.java:126 org.apache.poi.poifs.filesystem.NDocumentOutputStream.write()

```
124                     checkBufferSize();
125                 } else {
126                     write(b, 0, b.length);
127                 }
128             }
```

Analysis: Suspicious

ChunkedCipherOutputStream.java, line 81 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function write() in ChunkedCipherOutputStream.java might reveal system data or debugging information by calling write() on line 81. The information revealed by write() could help an adversary form a plan of attack.

Source: NPOIFSDocument.java:225 java.lang.Throwable.getMessage()

```
223                 }
224             } catch (IOException e) {
225                 result = e.getMessage();
226             }
227             results[0] = result;
```

Sink: ChunkedCipherOutputStream.java:81 org.apache.poi.poifs.crypt.ChunkedCipherOutputStream.write()

```
79
80          public void write(byte[] b) throws IOException {
81              write(b, 0, b.length);
82          }
```

Analysis: Suspicious

NDocumentOutputStream.java, line 102 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract: The function checkBufferSize() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 102. The information revealed by write() could help an adversary form a plan of attack.

Source: EscherBitmapBlip.java:129 Read e()

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink: NDocumentOutputStream.java:102 org.apache.poi.poifs.filesystem.NDocumentOutputStream.write()

```
100                     byte[] data = _buffer.toByteArray();
101                     _buffer = null;
102                     write(data, 0, data.length);
103                 } else {
104                     // So far, mini stream will work, keep going
```

Analysis: Suspicious

NDocumentOutputStream.java, line 126 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		

Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 126. The information revealed by write() could help an adversary form a plan of attack.
Source:	POIFSDocument.java:367 java.lang.Throwable.getMessage()

```
365                 }
366               } catch (IOException e) {
367                 result = e.getMessage();
368               }
369               results[0] = result;
```

Sink:	NDocumentOutputStream.java:126 org.apache.poi.poifs.filesystem.NDocumentOutputStream.write()

```
124                     checkBufferSize();
125                 } else {
126                     write(b, 0, b.length);
127                 }
128             }
```

Analysis:	Suspicious

ExcelFileFormatDocFunctionExtractor.java, line 487 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in ExcelFileFormatDocFunctionExtractor.java might reveal system data or debugging information by calling write() on line 487. The information revealed by write() could help an adversary form a plan of attack.		
Source:	EscherBitmapBlip.java:129 Read e()		

```
127                 catch ( Exception e )
128                 {
129                     extraData = e.toString();
130                 }
131                 StringBuilder builder = new StringBuilder();
```

Sink:	ExcelFileFormatDocFunctionExtractor.java:487 java.io.OutputStream.write()

```
485
486               }
487               _os.write(b, off, len);
488             }
489           }
```

Analysis:	Suspicious

MemoryPackagePartOutputStream.java, line 93 (System Information Leak: External)

Fortify Priority:	Low	**Folder**	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in MemoryPackagePartOutputStream.java might reveal system data or debugging information by calling write() on line 93. The information revealed by write() could help an adversary form a plan of attack.		
Source:	POIFSDocument.java:367 java.lang.Throwable.getMessage()		

```
365                 }
366               } catch (IOException e) {
367                 result = e.getMessage();
368               }
369               results[0] = result;
```

Sink:	MemoryPackagePartOutputStream.java:93 java.io.OutputStream.write()

```
91              @Override
92              public void write(byte[] b) throws IOException {
93                _buff.write(b);
94              }
95            }
```

Analysis:	Suspicious

MemoryPackagePartOutputStream.java, line 93 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in MemoryPackagePartOutputStream.java might reveal system data or debugging information by calling write() on line 93. The information revealed by write() could help an adversary form a plan of attack.		

Source: NPOIFSDocument.java:225 java.lang.Throwable.getMessage()

```
223                 }
224               } catch (IOException e) {
225                 result = e.getMessage();
226               }
227               results[0] = result;
```

Sink: MemoryPackagePartOutputStream.java:93 java.io.OutputStream.write()

```
91  @Override
92  public void write(byte[] b) throws IOException {
93    _buff.write(b);
94  }
95  }
```

Analysis: Suspicious

NDocumentOutputStream.java, line 123 (System Information Leak: External)

Fortify Priority:	Low	Folder	Low
Kingdom:	Encapsulation		
Abstract:	The function write() in NDocumentOutputStream.java might reveal system data or debugging information by calling write() on line 123. The information revealed by write() could help an adversary form a plan of attack.		

Source: POIFSDocument.java:367 java.lang.Throwable.getMessage()

```
365     }
366   } catch (IOException e) {
367     result = e.getMessage();
368   }
369   results[0] = result;
```

Sink: NDocumentOutputStream.java:123 java.io.OutputStream.write()

```
121
122             if (_buffer != null) {
123                 _buffer.write(b);
124                 checkBufferSize();
125             } else {
```

Analysis: Suspicious