| **Case 1** <https://github.com/koral--/android-gif-drawable/commit/9080ef9dd83a69d024e692f7fdf6647a99506bb2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/830> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/9080ef9dd83a69d024e692f7fdf6647a99506bb2>    Bug is at line 88 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  88(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  …  -> 84(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  -> 81(java\_GifDecoder\_18-cfg.dot) (seekToTime)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 2** <https://github.com/koral--/android-gif-drawable/commit/9080ef9dd83a69d024e692f7fdf6647a99506bb2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/830> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/9080ef9dd83a69d024e692f7fdf6647a99506bb2>    Bug is at line 88 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  116(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  …  -> 112(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  -> 93(java\_GifDecoder\_18-cfg.dot) (seekToFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 3** <https://github.com/koral--/android-gif-drawable/commit/7a700e9e0a563a06dc95414f1820ad8007db7e35> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/657> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/7a700e9e0a563a06dc95414f1820ad8007db7e35>    Bug is at line 66 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  66(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_getCurrentPosition)  …  -> 60(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_getCurrentPosition)  -> 265(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add ternary operator to init variable  Location: assignment statement + loop |

| **Case 4** <https://github.com/koral--/android-gif-drawable/commit/8610823158d5276eb1cedd766a4fbc13719798cd> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/797> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/8610823158d5276eb1cedd766a4fbc13719798cd>    Bug is at line 239 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  239(c\_dgif\_lib\_9-cfg.dot) (DGifGetImageDesc)  …  -> 217(c\_dgif\_lib\_9-cfg.dot) (DGifGetImageDesc)  -> 33(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 18(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression  Location: conditional exression |

| **Case 5** <https://github.com/koral--/android-gif-drawable/commit/57e58879a222f748a6f03413c50e77c32d96e1e1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/797> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/57e58879a222f748a6f03413c50e77c32d96e1e1>    Bug is at line 21 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  21(c\_init\_3-cfg.dot) (createGifInfo)  …  -> 3(c\_init\_3-cfg.dot) (createGifInfo)  -> 188(c\_decoding\_8-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openByteArray)  …  -> 163(c\_decoding\_8-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openByteArray)  -> 42(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewGlobalRef | Foreign function | Global and Local References | | GetArrayLength | Foreign function | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add function call to release memory  Location: if/branching statement + function call |

| **Case 6** <https://github.com/koral--/android-gif-drawable/commit/e46c42d04eb5126f060a67c8bd0d1c7519240c9f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/594> where it says    Thus, the symptom is no description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/e46c42d04eb5126f060a67c8bd0d1c7519240c9f>    Bug is at line 213 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  213(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openDirectByteBuffer)  …  -> 217(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openDirectByteBuffer)  -> 46(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewGlobalRef | Foreign function | Global and Local References | | GetArrayLength | Foreign function | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + function call |

| **Case 7** <https://github.com/koral--/android-gif-drawable/commit/c89f0b07c16d7aead63acf4b8762ec5b8a8abc3a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/570> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/c89f0b07c16d7aead63acf4b8762ec5b8a8abc3a>    Bug is at line 77 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  path:  77(java\_GifInfoHandle\_20-cfg.dot) (openFd)  …  -> 83(java\_GifInfoHandle\_20-cfg.dot) (openFd)  -> 327(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openNativeFileDescriptor)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the exception  Location: try-catch statement |

| **Case 8** <https://github.com/koral--/android-gif-drawable/commit/98a3cb6e83527dad814f02e68feb8291b503086f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/429> where it says    Thus, the symptom is hang/stuck |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/98a3cb6e83527dad814f02e68feb8291b503086f>    Bug is at line 226 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  226(java\_GifTextureView\_27-cfg.dot) (run)  …  -> 208(java\_GifTextureView\_27-cfg.dot) (run)  -> 28(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_setSpeedFactor)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * C ->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete release statement  Location: try-catch statement |

| **Case 9** <https://github.com/koral--/android-gif-drawable/commit/da8587ffa684d42d234a27b74f558cd7febfbb1b> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/797> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/da8587ffa684d42d234a27b74f558cd7febfbb1b>    Bug is at line 221 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  221(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openDirectByteBuffer)  …  -> 208(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openDirectByteBuffer)  -> 215(java\_GifInfoHandle\_20-cfg.dot) (setSpeedFactor)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support | | GetDirectBufferCapacity | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add global references to byte buffers  Location: assignment statement |

| **Case 10** <https://github.com/koral--/android-gif-drawable/commit/018b12fe5753a15b3b477983472a6d0913f046c9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/394> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/018b12fe5753a15b3b477983472a6d0913f046c9>    Bug is at line 26 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  26(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 17(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change return to break  Location: loop + if/branching statement |

| **Case 11** <https://github.com/koral--/android-gif-drawable/commit/16bb3c1e6abd23df5ee609b8ee14181f58e349df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/394> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/16bb3c1e6abd23df5ee609b8ee14181f58e349df>    Bug is at line 16 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  16(c\_dispose\_10-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_free)  …  -> 3(c\_dispose\_10-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_free)  -> 166(java\_GifInfoHandle\_20-cfg.dot) (recycle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetMethodID | Foreign | Calling Instance Methods | | CallVoidMethod | Foreign | Calling Instance Methods |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete if/branching statement  Location: foreign function + if/branching statement |

| **Case 12** <https://github.com/koral--/android-gif-drawable/commit/9063377dcca362ffc3a4e993791d2932dbc9b9fa> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/353> where it says    Thus, the symptom is segment fault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/9063377dcca362ffc3a4e993791d2932dbc9b9fa>    Bug is at line 67 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  67(c\_gif\_6-cfg.dot) (streamReadFun)  …  -> 54(c\_gif\_6-cfg.dot) (streamReadFun)  -> 276(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openStream)  …  -> 218(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openStream)  -> 53(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetByteArrayRegion | Foreign | Array Operations | | CallIntMethod | Foreign | Calling Instance Methods |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument variable  Location: foreign function + if/branching statement + assignment statement |

| **Case 13** <https://github.com/koral--/android-gif-drawable/commit/d71dffb2ca5301a07939f46aa3570ad3ef09b129> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/330> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/d71dffb2ca5301a07939f46aa3570ad3ef09b129>    Bug is at line 106 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  106(c\_drawing\_7-cfg.dot) (disposeFrameIfNeeded)  …  -> 87(c\_gif\_6-cfg.dot) (disposeFrameIfNeeded)  -> 142(c\_gif\_6-cfg.dot) (drawNextBitmap)  …  -> 140(c\_gif\_6-cfg.dot) (drawNextBitmap)  -> 78(c\_control\_14-cfg.dot) (seek)  …  -> 43(c\_control\_14-cfg.dot) (seek)  -> 38(c\_control\_14-cfg.dot) (seekBitmap)  …  -> 33(c\_control\_14-cfg.dot) (seekBitmap)  -> 107(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  …  -> 84(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  -> 53(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression  Location: conditional expression |

| **Case 14** <https://github.com/koral--/android-gif-drawable/commit/2d375e93093a3966293124782dfeff9f50d90ba5> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/koral--/android-gif-drawable/issues/309> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/2d375e93093a3966293124782dfeff9f50d90ba5>    Bug is at line 105 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  105(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 3(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 15** <https://github.com/koral--/android-gif-drawable/commit/5077341d580556db74759ae240457d59cd7fd70a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/305> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/5077341d580556db74759ae240457d59cd7fd70a>    Bug is at line 194 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  194(c\_surface\_16-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_bindSurface)  …  -> 56(c\_surface\_16-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_bindSurface)  -> 161(java\_GifInfoHandle\_20-cfg.dot) (bindSurface)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the operator  Location: assignment statement |

| **Case 16** <https://github.com/koral--/android-gif-drawable/commit/45b00645accccab60a545bfb38ded3571518420e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/297> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/45b00645accccab60a545bfb38ded3571518420e>    Bug is at line 23 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  23(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 3(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the code chunk location  Location: if/branching statement |

| **Case 17** <https://github.com/koral--/android-gif-drawable/commit/e84970b368af27fbfbe37ca46e3d948ac0668cc5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/276> where it says    Thus, the symptom is error message (runtime error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/e84970b368af27fbfbe37ca46e3d948ac0668cc5>    Bug is at line 59 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  59(c\_opengl\_19-cfg.dot) (releaseTexImageDescriptor)  …  -> 52(c\_opengl\_19-cfg.dot) (releaseTexImageDescriptor)  -> 104(c\_opengl\_19-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_startDecoderThread)  …  -> 76(c\_opengl\_19-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_startDecoderThread)  -> 297(java\_GifInfoHandle\_20-cfg.dot) (startDecoderThread)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the code chunk location  Location: if/branching statement |

| **Case 18** <https://github.com/koral--/android-gif-drawable/commit/a527450494d2dd07e8c9784bbbf6962e6dbc808c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/277> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/a527450494d2dd07e8c9784bbbf6962e6dbc808c>    Bug is at line 46 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  46(c\_control\_14-cfg.dot) (seek)  …  -> 44(c\_control\_14-cfg.dot) (seek)  -> 39(c\_control\_14-cfg.dot) (seekBitmap)  …  -> 34(c\_control\_14-cfg.dot) (seekBitmap)  -> 104(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToFrame)  …  -> 76(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToFrame)  -> 231(java\_GifInfoHandle\_20-cfg.dot) (seekToFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a new function  Location: if/branching statement |

| **Case 19** <https://github.com/koral--/android-gif-drawable/commit/f1d2a922b55699ea8e4a1b253d9f23185d08a2a7> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/koral--/android-gif-drawable/issues/275 where it says    Thus, the symptom is error message (runtime error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/f1d2a922b55699ea8e4a1b253d9f23185d08a2a7>    Bug is at line 89 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  89(c\_opengl\_19-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_startDecoderThread)  …  -> 51(c\_opengl\_19-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_startDecoderThread)  -> 297(java\_GifInfoHandle\_20-cfg.dot) (startDecoderThread)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression  Location: conditional expression |

| **Case 20** <https://github.com/koral--/android-gif-drawable/commit/c09f22408edb1e8c9d38b2cdc34b9f3b46e29714> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/250> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/c09f22408edb1e8c9d38b2cdc34b9f3b46e29714>    Bug is at line 32 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  32(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 18(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: assignment statement |

| **Case 21** <https://github.com/koral--/android-gif-drawable/commit/bf7b97d141c24323743b5aa68a386f166671fd9e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/240> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/bf7b97d141c24323743b5aa68a386f166671fd9e>    Bug is at line 213 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  213(c\_surface\_16-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_postUnbindSurface)  …  -> 206(c\_surface\_16-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_postUnbindSurface)  -> 231(java\_GifInfoHandle\_20-cfg.dot) (postUnbindSurface)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression  Location: conditional expression |

| **Case 22** <https://github.com/koral--/android-gif-drawable/commit/bd1dea98d72c999b764aee1f71fdb02c90f6a8dc> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/koral--/android-gif-drawable/issues/194> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/bd1dea98d72c999b764aee1f71fdb02c90f6a8dc>    Bug is at line 21 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  21(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 3(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression + add data processing statements  Location: conditional expression + if/branching statements |

| **Case 23** <https://github.com/koral--/android-gif-drawable/commit/c72f51d21d7fe55e14b45752531e80c62cc6495b> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/koral--/android-gif-drawable/issues/186> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/c72f51d21d7fe55e14b45752531e80c62cc6495b>    Bug is at line 92 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  92(c\_decoding\_11-cfg.dot) (DDGifSlurp)  …  -> 3(c\_decoding\_11-cfg.dot) (DDGifSlurp)  -> 213(c\_metadata\_15-cfg.dot) (restoreSavedState)  …  -> 189(c\_metadata\_15-cfg.dot) (restoreSavedState)  -> 238(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  …  -> 230(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreSavedState)  -> 319(java\_GifInfoHandle\_20-cfg.dot) (getCurrentPosition)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add data processing statements  Location: if/branching statement |

| **Case 24** <https://github.com/koral--/android-gif-drawable/commit/196676866b7de8aea07e2cbbd9ea084fa4cfe960> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/167> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/196676866b7de8aea07e2cbbd9ea084fa4cfe960>    Bug is at line 96 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  96(c\_drawing\_7-cfg.dot) (disposeFrameIfNeeded)  …  -> 85(c\_gif\_6-cfg.dot) (disposeFrameIfNeeded)  -> 142(c\_gif\_6-cfg.dot) (drawNextBitmap)  …  -> 140(c\_gif\_6-cfg.dot) (drawNextBitmap)  -> 78(c\_control\_14-cfg.dot) (seek)  …  -> 43(c\_control\_14-cfg.dot) (seek)  -> 38(c\_control\_14-cfg.dot) (seekBitmap)  …  -> 33(c\_control\_14-cfg.dot) (seekBitmap)  -> 107(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  …  -> 84(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  -> 53(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the function  Location: assignment statement + function call |

| **Case 25** <https://github.com/koral--/android-gif-drawable/commit/f5cc3b3f071324dc1735941eabc95c190771de47> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/154> where it says    Thus, the symptom is error message (runtime error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/f5cc3b3f071324dc1735941eabc95c190771de47>    Bug is at line 15 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  path:  15(java\_GifViewSavedState\_39-cfg.dot) (GifViewSavedState)  …  -> 19(java\_GifViewSavedState\_39-cfg.dot) (GifViewSavedState)  -> 117(c\_metadata\_15-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_getSavedState)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewLongArray | Foreign | Array Operations | | SetLongArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data handling error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change length of array  Location: assignment statement |

| **Case 26** <https://github.com/koral--/android-gif-drawable/commit/aaa29bbc2721c71f5b521242f3da40f878cbccdb> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/151> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/aaa29bbc2721c71f5b521242f3da40f878cbccdb>    Bug is at line 34 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  34(java\_RenderTask\_38-cfg.dot) (doWork)  …  -> 19(java\_RenderTask\_38-cfg.dot) (doWork)  -> 55(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * C-> Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression and location  Location: condition expression + if/branching statement |

| **Case 27** <https://github.com/koral--/android-gif-drawable/commit/790f9c7f5a34263799cc6153fd537e3fed7544ab> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/152> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/790f9c7f5a34263799cc6153fd537e3fed7544ab>    Bug is at line 124 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  34(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreRemainder)  …  -> 19(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_restoreRemainder)  -> 135(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression  Location: condition expression |

| **Case 28** <https://github.com/koral--/android-gif-drawable/commit/f1bed87f10dc31eab783faaa318b1bec10628a9d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/150> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/f1bed87f10dc31eab783faaa318b1bec10628a9d>    Bug is at line 62 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  62(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  …  -> 51(c\_control\_14-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_seekToTime)  -> 189(java\_GifInfoHandle\_20-cfg.dot) (seekToTime)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression  Location: condition expression |

| **Case 29** <https://github.com/koral--/android-gif-drawable/commit/53c40a7a718081d0f1ad4764d1cd64c67b4b768c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/122> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/53c40a7a718081d0f1ad4764d1cd64c67b4b768c>    Bug is at line 16 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  16(c\_bitmap\_5-cfg.dot) (lockPixels)  …  -> 4(c\_bitmap\_5-cfg.dot) (lockPixels)  ->77(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  …  -> 55(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  -> 114(java\_GifInfoHandle\_20-cfg.dot) (renderFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete loop and sleep function  Location: loop + if/branching statement |

| **Case 30** <https://github.com/koral--/android-gif-drawable/commit/53c40a7a718081d0f1ad4764d1cd64c67b4b768c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/122> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/53c40a7a718081d0f1ad4764d1cd64c67b4b768c>    Bug is at line 6 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  6(c\_bitmap\_5-cfg.dot) (lockPixels)  …  -> 4(c\_bitmap\_5-cfg.dot) (lockPixels)  ->77(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  …  -> 55(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  -> 114(java\_GifInfoHandle\_20-cfg.dot) (renderFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 31** <https://github.com/koral--/android-gif-drawable/commit/2a70d0abae2ac568589ebf51330e6d52a719acca> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/122> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/2a70d0abae2ac568589ebf51330e6d52a719acca>    Bug is at line 618 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  611(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFd)  …  -> 588(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFd)  ->88 (java\_GifInfoHandle\_20-cfg.dot) (openAssetFileDescriptor)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetFieldID | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add ternary operator to assign value  Location: assignment statement + return |

| **Case 32** <https://github.com/koral--/android-gif-drawable/commit/5aece1ab599d4cd0e92eb823640f722ee6e18e11> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/125> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/5aece1ab599d4cd0e92eb823640f722ee6e18e11>    Bug is at line 270 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  270(c\_gif\_6-cfg.dot) (readExtensions)  …  -> 230(c\_gif\_6-cfg.dot) (readExtensions)  ->352(c\_gif\_6-cfg.dot) (DDGifSlurp)  …  -> 271(c\_gif\_6-cfg.dot) (DDGifSlurp)  ->454(c\_gif\_6-cfg.dot) (createGifHandle)  …  -> 398(c\_bitmap\_5-cfg.dot) (createGifHandle)  ->500(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFile)  …  -> 484(c\_bitmap\_5-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFile)  -> 114(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression  Location: conditional expression |

| **Case 33** <https://github.com/koral--/android-gif-drawable/commit/3d632ce579c6fd7fd4eca24a889a874b98f0eefe> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/114> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/3d632ce579c6fd7fd4eca24a889a874b98f0eefe>    Bug is at line 959 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  959(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  …  -> 907(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  -> 95(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 34** <https://github.com/koral--/android-gif-drawable/commit/c0da2dfdb3244948d261229288767f6d65cb109c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/105> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/c0da2dfdb3244948d261229288767f6d65cb109c>    Bug is at line 76 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  76(c\_gif\_6-cfg.dot) (cleanUp)  …  -> 59(c\_gif\_6-cfg.dot) (cleanUp)  ->468(c\_gif\_6-cfg.dot) (createGifHandle)  …  -> 427(c\_bitmap\_5-cfg.dot) (createGifHandle)  ->529(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFile)  …  -> 513(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openFile)  -> 52(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 35** <https://github.com/koral--/android-gif-drawable/commit/709838234c5f6f5e4e7359d815348a1a26e1a571> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/105> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/709838234c5f6f5e4e7359d815348a1a26e1a571>    Bug is at line 83 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  path:  83(java\_GifInfoHandle\_20-cfg.dot) (run)  …  -> 78(java\_GifInfoHandle\_20-cfg.dot) (run)  -> 33(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 36** <https://github.com/koral--/android-gif-drawable/commit/635503517b5483bdb3d7f02e421992da0b5f3c12> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/101> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/635503517b5483bdb3d7f02e421992da0b5f3c12>    Bug is at line 114 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  114(c\_dgif\_lib\_12-cfg.dot) (DGifOpen)  …  -> 47(c\_dgif\_lib\_12-cfg.dot) (DGifOpen)  ->185(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openByteArray)  …  -> 163(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_openByteArray)  -> 44(java\_GifInfoHandle\_20-cfg.dot) (GifInfoHandle)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewGlobalRef | Foreign | Global and Local References | | GetArrayLength | Foreign | Array Operations | | DeleteGlobalRef | Foreign | Global and Local References |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: initialize a variable  Location: assignment statement |

| **Case 37** <https://github.com/koral--/android-gif-drawable/commit/93171914646ed9f04960018f0b9d38d66349cdd9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/98> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/93171914646ed9f04960018f0b9d38d66349cdd9>    Bug is at line 114 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  114(java\_GifDrawable\_36-cfg.dot) (run)  …  -> 70(java\_GifDrawable\_36-cfg.dot) (run)  -> 909(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifInfoHandle\_renderFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * C-> Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument of function call + add a function call statement  Location: function call |

| **Case 38** <https://github.com/koral--/android-gif-drawable/commit/349d3e2c58533ea3112d7373f21ef69f3d25eaee> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/81> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  This modification ensures that when converting a byte type value to a short or a larger width type, the issue of sign extension is avoided.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/349d3e2c58533ea3112d7373f21ef69f3d25eaee>    Bug is at line 348 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  path:  348(c\_gif\_6-cfg.dot) (readExtensions)  …  -> 339(c\_gif\_6-cfg.dot) (readExtensions)  -> 489(c\_gif\_6-cfg.dot) (DDGifSlurp)  …  -> 388(c\_gif\_6-cfg.dot) (DDGifSlurp)  -> 960(c\_gif\_6-cfg.dot) (getBitmap)  …  -> 952(c\_gif\_6-cfg.dot) (getBitmap)  -> 1042(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 1008(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 516(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: do bitwise operations  Location: assignment statement |

| **Case 39** <https://github.com/koral--/android-gif-drawable/commit/832ca2a1dee3b93c80484978e0ee6a2fc345c7dc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/80> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/832ca2a1dee3b93c80484978e0ee6a2fc345c7dc>    Bug is at line 959 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  959(c\_gif\_6-cfg.dot) (getBitmap)  …  -> 950(c\_gif\_6-cfg.dot) (getBitmap)  -> 1042(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 1008(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 516(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 40** <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/68> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10>    Bug is at line 1031 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1042(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 1008(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 513(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 41** <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/68> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10>    Bug is at line 527 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  527(c\_gif\_6-cfg.dot) (setMetaData)  …  ->1008(c\_gif\_6-cfg.dot) (setMetaData)  ->637(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openFile)  …  ->631(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openFile)  ->189(java\_GifDrawable\_36-cfg.dot) (GifDrawable)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 42** <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/68> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10>    Bug is at line 1064 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1064(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToFrame)  …  -> 1042(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToFrame)  -> 513(java\_GifDrawable\_36-cfg.dot) (seekToFrame)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 43** <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/68> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/1f4e9cac18ffb19da9a600b13b5d3e4951da5b10>    Bug is at line 1097 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1097(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_renderFrame)  …  -> 1076(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_renderFrame)  -> 513(java\_GifDrawable\_36-cfg.dot) (draw)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetMethodID | Foreign | Calling Instance Methods | | CallVoidMethod | Foreign | Calling Instance Methods | | ExceptionOccurred | Foreign | Exceptions | | ExceptionClear | Foreign | Exceptions | | DeleteGlobalRef | Foreign | Global and Local References |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 44** <https://github.com/koral--/android-gif-drawable/commit/0024f0e7e7127cff17d3583e83fda8da5855fce6> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/70> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/0024f0e7e7127cff17d3583e83fda8da5855fce6>    Bug is at line 481 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  481(c\_gif\_6-cfg.dot) (DDGifSlurp)  …  -> 385(c\_gif\_6-cfg.dot) (DDGifSlurp)  -> 960(c\_gif\_6-cfg.dot) (getBitmap)  …  -> 952(c\_gif\_6-cfg.dot) (getBitmap)  -> 1042(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 1008(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 516(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 45** <https://github.com/koral--/android-gif-drawable/commit/946ff557d99ac0805401ac164fe97be8213a0d9f> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/koral--/android-gif-drawable/issues/60> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/946ff557d99ac0805401ac164fe97be8213a0d9f>    Bug is at line 1109 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1109(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_renderFrame)  …  -> 1074(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_renderFrame)  -> 702(java\_GifDrawable\_36-cfg.dot) (draw)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 46** <https://github.com/koral--/android-gif-drawable/commit/a10a0abfe41786e6ca9ccd9272bc7b6a7447b707> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/56> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  The assignment from b[3] to transpIndex is safe and not affected by the sign of the original char value. transpIndex always receives an unsigned integer value within the expected range, regardless of the implementation of char as signed or unsigned across different compilers and platforms.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/a10a0abfe41786e6ca9ccd9272bc7b6a7447b707>    Bug is at line 351 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  351(c\_gif\_6-cfg.dot) (readExtensions)  …  -> 338(c\_gif\_6-cfg.dot) (readExtensions)  -> 484(c\_gif\_6-cfg.dot) (DDGifSlurp)  …  -> 385(c\_gif\_6-cfg.dot) (DDGifSlurp)  -> 943(c\_gif\_6-cfg.dot) (getBitmap)  …  -> 937(c\_gif\_6-cfg.dot) (getBitmap)  -> 1029(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 996(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 516(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Conversion error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Changing the cast type to a bitwise operation  Location: if/branching statement |

| **Case 47** <https://github.com/koral--/android-gif-drawable/commit/f0097fddae876c03d9b8897c256492738619ee0a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/52> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/f0097fddae876c03d9b8897c256492738619ee0a>    Bug is at line 989 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  989(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 984(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 516(java\_GifDrawable\_36-cfg.dot) (seekTo)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 48** <https://github.com/koral--/android-gif-drawable/commit/f0097fddae876c03d9b8897c256492738619ee0a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/52> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/f0097fddae876c03d9b8897c256492738619ee0a>    Bug is at line 1033 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1033(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  …  -> 1028(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_seekToTime)  -> 544(java\_GifDrawable\_36-cfg.dot) (seekToFrame )  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetIntArrayElements | Foreign | Array Operations | | ReleaseIntArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 49** <https://github.com/koral--/android-gif-drawable/commit/d13b649d3aed438ef9acde3b770b68f52ced494e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/50> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/d13b649d3aed438ef9acde3b770b68f52ced494e>    Bug is at line 590 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  590(c\_gif\_6-cfg.dot) (open)  …  -> 537(c\_gif\_6-cfg.dot) (open)  -> 637(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openFile)  …  -> 615(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openFile)  -> 181(java\_GifDrawable\_36-cfg.dot) (GifDrawable)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change code order  Location: if/branching statement |

| **Case 50** <https://github.com/koral--/android-gif-drawable/commit/cd0e4e61d1362f034065b7ddad892a1525fedfe1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/koral--/android-gif-drawable/issues/38> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/koral--/android-gif-drawable/commit/cd0e4e61d1362f034065b7ddad892a1525fedfe1>    Bug is at line 650 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  650(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openByteArray)  …  -> 632(c\_gif\_6-cfg.dot) (Java\_pl\_droidsonroids\_gif\_GifDrawable\_openByteArray)  -> 262(java\_GifDrawable\_36-cfg.dot) (GifDrawable)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewGlobalRef | Foreign | Global and Local References | | GetArrayLength | Foreign | Array Operations | | DeleteGlobalRef | Foreign | Global and Local References |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + free buffer  Location: if/branching statement + return + foreign function |

| **Case 51** <https://github.com/PojavLauncherTeam/PojavLauncher/commit/ee1c304a32a61a830373ed78a29be9cc53116b05> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/PojavLauncherTeam/PojavLauncher/issues/3113> where it says    Thus, the symptom is installation error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/PojavLauncherTeam/PojavLauncher/commit/ee1c304a32a61a830373ed78a29be9cc53116b05>    Bug is at line 274 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  274(java\_JavaGUILauncherActivity\_5-cfg.dot) (launchJavaRuntime)  -> 191(c\_stdio\_is\_20-cfg.dot) (Java\_net\_kdt\_pojavlaunch\_Logger\_appendToLog)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFLength | Foreign | String Operations | | GetStringUTFRegion | Foreign | String Operations | | GetStringLength | Foreign | String Operations | | CallVoidMethod | Foreign | Calling Instance Methods |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change code order + Boundary check against value range/limit  Location: if/branching statement + function call |

| **Case 51** <https://github.com/PojavLauncherTeam/PojavLauncher/commit/ee1c304a32a61a830373ed78a29be9cc53116b05> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/PojavLauncherTeam/PojavLauncher/issues/3113> where it says    Thus, the symptom is installation error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/PojavLauncherTeam/PojavLauncher/commit/ee1c304a32a61a830373ed78a29be9cc53116b05>    Bug is at line 274 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  274(java\_JavaGUILauncherActivity\_5-cfg.dot) (launchJavaRuntime)  -> 191(c\_stdio\_is\_20-cfg.dot) (Java\_net\_kdt\_pojavlaunch\_Logger\_appendToLog)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFLength | Foreign | String Operations | | GetStringUTFRegion | Foreign | String Operations | | GetStringLength | Foreign | String Operations | | CallVoidMethod | Foreign | Calling Instance Methods |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change code order + Boundary check against value range/limit  Location: if/branching statement + function call |

| **Case 52** <https://github.com/PojavLauncherTeam/PojavLauncher/commit/1b507bb5833f78bab0af30d6404e4caf59d4865a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/PojavLauncherTeam/PojavLauncher/issues/619> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/PojavLauncherTeam/PojavLauncher/commit/1b507bb5833f78bab0af30d6404e4caf59d4865a>    Bug is at line 72 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  72(java\_Tools\_6-cfg.dot) (launchMinecraft)  …  102(java\_Tools\_6-cfg.dot) (launchMinecraft)  276(java\_JREUtils\_7-cfg.dot) (launchJavaVM)  ..  312(java\_JREUtils\_7-cfg.dot) (launchJavaVM)  -> 191(c\_stdio\_is\_20-cfg.dot) (Java\_net\_kdt\_pojavlaunch\_Logger\_appendToLog)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFLength | Foreign | String Operations | | GetStringUTFRegion | Foreign | String Operations | | GetStringLength | Foreign | String Operations | | CallVoidMethod | Foreign | Calling Instance Methods |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 53** <https://github.com/LWJGL/lwjgl3/commit/6ff21eaf542ab397ce58f069ed2cb8656f927f29> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/PojavLauncherTeam/PojavLauncher/issues/619> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/6ff21eaf542ab397ce58f069ed2cb8656f927f29>    Bug is at line 112 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  112(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_invokeD\_\_J)  …  -> 110(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_invokeD\_\_J)  -> 191(java\_stdio\_is\_20-cfg.dot) (Java\_net\_kdt\_pojavlaunch\_Logger\_appendToLog)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Build error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the function type  Location: return statement |

| **Case 54** <https://github.com/LWJGL/lwjgl3/commit/e19a2f6bfed5f333f8a01009240f1fff5e0f488f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/755> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/e19a2f6bfed5f333f8a01009240f1fff5e0f488f>    Bug is at line 760 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  760(java\_vulkan/EXTDebugUtils\_21-cfg.dot) (nvkCmdInsertDebugUtilsLabelEXT)  …  -> 765(java\_vulkan/EXTDebugUtils\_21-cfg.dot) (nvkCmdInsertDebugUtilsLabelEXT)  -> 2362(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_callPP\_\_JJ)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change function  Location: function call + assignment |

| **Case 55** <https://github.com/LWJGL/lwjgl3/commit/0db2149b74a1300d65d7e0ebdf2b5aaaa648a954> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/720> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/0db2149b74a1300d65d7e0ebdf2b5aaaa648a954>    Bug is at line 19 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  19(c\_org\_lwjgl\_system\_linux\_DynamicLinkLoader\_8-cfg.dot) (Java\_org\_lwjgl\_system\_linux\_DynamicLinkLoader\_ndlerror)  …  -> 17(c\_org\_lwjgl\_system\_linux\_DynamicLinkLoader\_8-cfg.dot) (Java\_org\_lwjgl\_system\_linux\_DynamicLinkLoader\_ndlerror)  -> 101(java\_DynamicLinkLoader\_22-cfg.dot) (dlerror)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change data type  Location: return statement |

| **Case 56** <https://github.com/LWJGL/lwjgl3/commit/ce56fc6c805ea671a4cf4e79abbd331688adfb6d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/673> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/ce56fc6c805ea671a4cf4e79abbd331688adfb6d>    Bug is at line 151 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  151(java\_KHRCreateRenderpass2\_23-cfg.dot) (nvkCmdBeginRenderPass2KHR)  …  -> 153(java\_KHRCreateRenderpass2\_23-cfg.dot) (nvkCmdBeginRenderPass2KHR)  -> 101(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_callPPPV\_\_JJJJ)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete function call  Location: function call |

| **Case 57** <https://github.com/LWJGL/lwjgl3/commit/ce56fc6c805ea671a4cf4e79abbd331688adfb6d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/673> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/ce56fc6c805ea671a4cf4e79abbd331688adfb6d>    Bug is at line 151 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  95(java\_KHRDisplaySwapchain\_24-cfg.dot) (nvkCreateSharedSwapchainsKHR)  …  -> 98(java\_KHRDisplaySwapchain\_24-cfg.dot) (nvkCreateSharedSwapchainsKHR)  -> 101(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_callPPPV\_\_JJJJ)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete function call  Location: function call |

| **Case 58** <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/623> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0>    Bug is at line 16 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  16(c\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_8-cfg.dot) (Java\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_ntinyfd\_1version)  …  -> 14(c\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_8-cfg.dot) (Java\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_ntinyfd\_1version)  -> 56(java\_TinyFileDialogs\_JNI\_25-cfg.dot) (tinyfd\_version)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change return statement  Location:return |

| **Case 59** <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/623> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0>    Bug is at line 21 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  21(c\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_8-cfg.dot) (Java\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_ntinyfd\_1needs)  …  -> 19(c\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_8-cfg.dot) (Java\_org\_lwjgl\_util\_tinyfd\_TinyFileDialogs\_ntinyfd\_1needs)  -> 69(java\_TinyFileDialogs\_JNI\_25-cfg.dot) (tinyfd\_needs)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change return statement  Location:return |

| **Case 60** <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/596> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/0b42c297795618d5f2ca6627a5a02140aeefe8e0>    Bug is at line 945 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  945(c\_stb\_vorbis\_9-cfg.dot) (setup\_malloc)  …  -> 935(c\_stb\_vorbis\_9-cfg.dot) (setup\_malloc)  -> 1274(c\_stb\_vorbis\_9-cfg.dot) (init\_blocksize)  …  -> 1271(c\_stb\_vorbis\_9-cfg.dot) (init\_blocksize)  -> 4124(c\_stb\_vorbis\_9-cfg.dot) (start\_decoder)  …  -> 3564(c\_stb\_vorbis\_9-cfg.dot) (start\_decoder)  -> 4505(c\_stb\_vorbis\_9-cfg.dot) (stb\_vorbis\_open\_pushdata)  …  -> 4495(c\_stb\_vorbis\_9-cfg.dot) (stb\_vorbis\_open\_pushdata)  -> 58(c\_org\_lwjgl\_stb\_STBVorbis\_10-cfg.dot) (Java\_org\_lwjgl\_stb\_STBVorbis\_nstb\_1vorbis\_1open\_1pushdata\_\_JIJJJ)  …  -> 52(c\_org\_lwjgl\_stb\_STBVorbis\_10-cfg.dot) (Java\_org\_lwjgl\_stb\_STBVorbis\_nstb\_1vorbis\_1open\_1pushdata\_\_JIJJJ)  -> 253(java\_STBVorbis\_26-cfg.dot) (stb\_vorbis\_open\_pushdata)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change return statement  Location: return |

| **Case 61** <https://github.com/LWJGL/lwjgl3/commit/30db58be70cc9cd8b62126e9d4d58b875a3c0ae7> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/596> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/30db58be70cc9cd8b62126e9d4d58b875a3c0ae7>    Bug is at line 1028 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  253(java\_CL12\_27-cfg.dot) (nclLinkProgram)  -> 2322(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_callPPPPPPPP\_\_JIJJIJJJJJ)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement |

| **Case 62** <https://github.com/LWJGL/lwjgl3/commit/50869cf9638ef2225a5ca141bf89d617a920a762> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/496> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/50869cf9638ef2225a5ca141bf89d617a920a762>    Bug is at line 2651 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  2651(java\_ClangIndex\_28-cfg.dot) (nclang\_getCString)  -> 298(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_invokePP\_\_JJ)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement |

| **Case 63** <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/490> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90>    Bug is at line 36 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  36(c\_org\_lwjgl\_bullet\_PhysicsClient\_11-cfg.dot) (Java\_org\_lwjgl\_bullet\_PhysicsClient\_nb3ConnectPhysicsTCP)  -> 1059(java\_PhysicsClient\_29-cfg.dot) (b3ConnectPhysicsTCP)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement |

| **Case 64** <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/490> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90>    Bug is at line 45 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  45(c\_org\_lwjgl\_bullet\_PhysicsClient\_11-cfg.dot) (Java\_org\_lwjgl\_bullet\_PhysicsClient\_nb3ConnectPhysicsUDP)  -> 1086(java\_PhysicsClient\_29-cfg.dot) (nb3ConnectPhysicsUDP)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement |

| **Case 65** <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/490> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/a3c56f712a5d79e632367106daa9e91b4da0ae90>    Bug is at line 54 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  54(c\_org\_lwjgl\_bullet\_PhysicsClient\_11-cfg.dot) (Java\_org\_lwjgl\_bullet\_PhysicsClient\_nb3DisconnectSharedMemory)  -> 1112(java\_PhysicsClient\_29-cfg.dot) (b3DisconnectSharedMemory)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement |

| **Case 66** <https://github.com/LWJGL/lwjgl3/commit/6775dc8e5fb8842551fe158d7403843a1830b7e5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/482> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/6775dc8e5fb8842551fe158d7403843a1830b7e5>    Bug is at line 144 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  54(c\_org\_lwjgl\_util\_lmdb\_LMDB\_12-cfg.dot) (JavaCritical\_org\_lwjgl\_util\_lmdb\_LMDB\_nmdb\_1env\_1set\_1mapsize)  -> 943(java\_LMDB\_30-cfg.dot) (mdb\_env\_set\_mapsize)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: return statement |

| **Case 67** <https://github.com/LWJGL/lwjgl3/commit/6775dc8e5fb8842551fe158d7403843a1830b7e5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/482> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/6775dc8e5fb8842551fe158d7403843a1830b7e5>    Bug is at line 474 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  474(c\_org\_lwjgl\_util\_lmdb\_LMDB\_12-cfg.dot) (JavaCritical\_org\_lwjgl\_util\_lmdb\_LMDB\_nmdb\_1cursor\_1count)  -> 1925(java\_LMDB\_30-cfg.dot) (mdb\_cursor\_count)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: assignment statement |

| **Case 68** <https://github.com/LWJGL/lwjgl3/commit/1ae772163cf3eb432cf6f327db5bf5b7fe9aad5f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/418> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/1ae772163cf3eb432cf6f327db5bf5b7fe9aad5f>    Bug is at line 75 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  75(java\_VRCompositor\_31-cfg.dot) (VRCompositor\_WaitGetPoses)  -> 60(java\_VRCompositor\_31-cfg.dot) (nVRCompositor\_WaitGetPoses)  …  -> 65(java\_VRCompositor\_31-cfg.dot) (nVRCompositor\_WaitGetPoses)  -> 1650(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot) (Java\_org\_lwjgl\_system\_JNI\_callPPI\_\_JIJ\_3I)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ReleasePrimitiveArrayCritical | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Argument error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument in function call  Location: return statement + function call |

| **Case 69** <https://github.com/LWJGL/lwjgl3/commit/94f8bd69e176be3e48b4376281fed778e76ab8df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/409> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/94f8bd69e176be3e48b4376281fed778e76ab8df>    Bug is at line 126 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  126(java\_MemoryTextDecoding\_32-cfg.dot) (decodeUTF16)  -> 730(java\_MemoryUtil\_33-cfg.dot) (MemoryUtil)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 103(c\_org\_lwjgl\_system\_MemoryJNI\_13-cfg.dot) (Java\_org\_lwjgl\_system\_MemoryAccessJNI\_ngetLong)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement + function call |

| **Case 70** <https://github.com/LWJGL/lwjgl3/commit/94f8bd69e176be3e48b4376281fed778e76ab8df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/409> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/94f8bd69e176be3e48b4376281fed778e76ab8df>    Bug is at line 30 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  30(java\_MemoryTextDecoding\_32-cfg.dot) (decodeASCII)  -> 730(java\_MemoryUtil\_33-cfg.dot) (MemoryUtil)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 103(c\_org\_lwjgl\_system\_MemoryJNI\_13-cfg.dot) (Java\_org\_lwjgl\_system\_MemoryAccessJNI\_ngetLong)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: return statement + function call |

| **Case 71** <https://github.com/LWJGL/lwjgl3/commit/f70df90fbb088cf66585ba4d3ebcf2c3fc4c843e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/276> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/f70df90fbb088cf66585ba4d3ebcf2c3fc4c843e>    Bug is at line 159 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  159(java\_MemoryTextDecoding\_32-cfg.dot) (setCapabilities)  -> 742(java\_MemoryUtil\_33-cfg.dot) (memAddress)  -> 730(java\_MemoryUtil\_33-cfg.dot) (memAddress0)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 103(c\_org\_lwjgl\_system\_MemoryJNI\_13-cfg.dot) (Java\_org\_lwjgl\_system\_MemoryAccessJNI\_ngetLong)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a function call  Location: function call |

| **Case 72** <https://github.com/LWJGL/lwjgl3/commit/f70df90fbb088cf66585ba4d3ebcf2c3fc4c843e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/276> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/f70df90fbb088cf66585ba4d3ebcf2c3fc4c843e>    Bug is at line 159 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  159(java\_MemoryTextDecoding\_32-cfg.dot) (setCapabilities)  -> 742(java\_MemoryUtil\_33-cfg.dot) (memAddress)  -> 730(java\_MemoryUtil\_33-cfg.dot) (memAddress0)  -> 138(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  …  -> 142(java\_MemoryAccessJNI\_34-cfg.dot) (getLong)  -> 103(c\_org\_lwjgl\_system\_MemoryJNI\_13-cfg.dot) (Java\_org\_lwjgl\_system\_MemoryAccessJNI\_ngetLong)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a function call  Location: function call |

| **Case 73** <https://github.com/LWJGL/lwjgl3/commit/d39d2f48ff9adc71b83fbdf3234ec81691aebd53> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/161> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/d39d2f48ff9adc71b83fbdf3234ec81691aebd53>    Bug is at line 159 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  159(java\_EGL\_33-cfg.dot) (createClientCapabilities)  …  -> 164(java\_EGL\_33-cfg.dot) (createClientCapabilities)  -> 174(c\_org\_lwjgl\_system\_JNI\_7-cfg.dot)(Java\_org\_lwjgl\_system\_JNI\_invokeI\_\_J)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Argument error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument variable  Location: function call + assignment statement |

| **Case 74** <https://github.com/LWJGL/lwjgl3/commit/4fb6618bfd8990b8a0b805f589a76ff4a1b56684> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/161> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/4fb6618bfd8990b8a0b805f589a76ff4a1b56684>    Bug is at line 24 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  24(java\_Pointer\_34-cfg.dot) (Pointer)  ->10(c\_org\_lwjgl\_system\_MemoryUtil\_14-cfg.dot)(Java\_org\_lwjgl\_system\_MemoryUtil\_memGetPointerSize)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: function call + assignment statement |

| **Case 75** <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/LWJGL/lwjgl3/issues/87> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e>    Bug is at line 24 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  177(java\_MemoryMnage\_35-cfg.dot) (realloc)  176(java\_MemoryMnage\_35-cfg.dot) (realloc)  …  72(java\_MemoryMnage\_35-cfg.dot) (realloc)  ->38(c\_org\_lwjgl\_system\_libc\_LibCStdlib\_15-cfg.dot)(Java\_org\_lwjgl\_system\_libc\_LibCStdlib\_nrealloc)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 75** <https://github.com/espeak-ng/espeak-ng/commit/55b2e9b2a3cf80989a4dcfe6afde122e3bb92297> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/55b2e9b2a3cf80989a4dcfe6afde122e3bb92297> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/55b2e9b2a3cf80989a4dcfe6afde122e3bb92297>    Bug is at line 1808 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1808(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1579(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Buffer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add an assignment to manage the value  Location: assignment |

| **Case 76** <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/55b2e9b2a3cf80989a4dcfe6afde122e3bb92297> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e>    Bug is at line 1859 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1859(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1579(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Buffer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add an assignment to manage the value  Location: assignment |

| **Case 77** <https://github.com/espeak-ng/espeak-ng/commit/e87691ad3b0d7e5fe3be5fe794413b7b86da2993> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/e87691ad3b0d7e5fe3be5fe794413b7b86da2993> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/e87691ad3b0d7e5fe3be5fe794413b7b86da2993>    Bug is at line 2669 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  2669(c\_dictionary\_25-cfg.dot) (LookupDict2)  …  ->2570(c\_dictionary\_25-cfg.dot) (LookupDict2)  ->2873(c\_dictionary\_25-cfg.dot) (LookupDictList)  …  ->2843(c\_dictionary\_25-cfg.dot) (LookupDictList)  ->2982(c\_dictionary\_25-cfg.dot) (Lookup)  …  ->2970(c\_dictionary\_25-cfg.dot) (Lookup)  ->2259(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2185(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a condition to check non-zero (true) to the loop continuation condition  Location: conditional expression |

| **Case 78** <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/55b2e9b2a3cf80989a4dcfe6afde122e3bb92297> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/LWJGL/lwjgl3/commit/e4bbe2db87f2d2d375c37f3f48020d3aa651067e>    Bug is at line 2084 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  2084(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1579(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Buffer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add an assignment to manage the value  Location: assignment |

| **Case 79** <https://github.com/espeak-ng/espeak-ng/commit/4aeb78e788b78a4bec6fa02f3ea04a84c035ccbc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/4aeb78e788b78a4bec6fa02f3ea04a84c035ccbc> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/4aeb78e788b78a4bec6fa02f3ea04a84c035ccbc>    Bug is at line 2084 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  543(c\_voices\_26-cfg.dot) (LoadVoice)  …  ->518(c\_voices\_26-cfg.dot) (LoadVoice)  ->1020(c\_voices\_26-cfg.dot) (LoadVoiceVariant)  …  ->1008(c\_voices\_26-cfg.dot) (LoadVoiceVariant)  ->1583(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: assign a value when intialize the variable  Location: assignment statement |

| **Case 80** <https://github.com/espeak-ng/espeak-ng/commit/b457d6f58ef645e2cab812a054a4c288b5139f38> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/b457d6f58ef645e2cab812a054a4c288b5139f38> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/b457d6f58ef645e2cab812a054a4c288b5139f38>    Bug is at line 259 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  259(c\_tr\_languages\_27-cfg.dot) (NewTranslator)  …  ->208(c\_tr\_languages\_27-cfg.dot) (NewTranslator)  ->482(c\_tr\_languages\_27-cfg.dot) (SelectTranslator)  …  ->464(c\_tr\_languages\_27-cfg.dot) (SelectTranslator)  ->276(c\_translateword\_26-cfg.dot) (SetAlternateTranslator)  …  ->263(c\_translateword\_26-cfg.dot) (SetAlternateTranslator)  ->428(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->302(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->922(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add initialization statements  Location: assignment statement |

| **Case 81** <https://github.com/espeak-ng/espeak-ng/commit/1ce5a1bb0b229084c7c5f5154d01c35795d69e3f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/1ce5a1bb0b229084c7c5f5154d01c35795d69e3f> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/1ce5a1bb0b229084c7c5f5154d01c35795d69e3f>    Bug is at line 705 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  705(c\_dictionary\_25-cfg.dot) (IsLetterGroup)  …  ->672(c\_dictionary\_25-cfg.dot) (IsLetterGroup)  ->1779(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1579(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Buffer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + add a loop  Location: if/branching statement + loop |

| **Case 82** <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559>    Bug is at line 381 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  381(c\_translate\_24-cfg.dot) (addPluralSuffixes)  …  ->379(c\_translate\_24-cfg.dot) (addPluralSuffixes)  ->684(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument  Location: assignment statement |

| **Case 83** <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559>    Bug is at line 95 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  95(c\_fifo\_28-cfg.dot) (fifo\_init)  …  ->72(c\_fifo\_28-cfg.dot) (fifo\_init)  ->277(c\_speech\_22-cfg.dot)(check\_data\_path)  …  ->247(c\_speech\_22-cfg.dot)(check\_data\_path)  ->308(c\_speech\_22-cfg.dot)(espeak\_ng\_InitializePath)  …  ->306(c\_speech\_22-cfg.dot)(espeak\_ng\_InitializePath)  ->53(c\_espeak\_api\_21-cfg.dot)(espeak\_Initialize)  …  ->51(c\_espeak\_api\_21-cfg.dot)(espeak\_Initialize)  ->158(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeCreate)  …  ->150(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeCreate)  ->287(java\_SpeechSynthesis\_1-cfg.dot)(attemptInit)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: assignment statement |

| **Case 84** <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559>    Bug is at line 290 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  290(c\_event\_29-cfg.dot) (polling\_thread)  …  ->253(c\_event\_29-cfg.dot) (polling\_thread)  ->94(c\_event\_29-cfg.dot)(event\_init)  …  ->78(c\_event\_29-cfg.dot)(event\_init)  ->139(c\_speech\_22-cfg.dot)(dispatch\_audio)  …  ->100(c\_speech\_22-cfg.dot)(dispatch\_audio)  ->211(c\_speech\_22-cfg.dot)(create\_events)  …  ->194(c\_speech\_22-cfg.dot)(create\_events)  ->470(c\_speech\_22-cfg.dot)(Synthesize)  …  ->426(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:  Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: assignment statement |

| **Case 85** <https://github.com/espeak-ng/espeak-ng/commit/c5355ed94e872eb8f248fadb22f7bab5fbd57201> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/issues/69> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/c5355ed94e872eb8f248fadb22f7bab5fbd57201>    Bug is at line 2455 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  2455(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->2177(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 86** <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/005a590786de16f74171c5d0aaca82d522774559>    Bug is at line 1167 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1167(c\_synthesize\_23-cfg.dot)(Generate)  …  ->1131(c\_synthesize\_23-cfg.dot)(Generate)  ->1581(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1531(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: assignment statement |

| **Case 87** <https://github.com/espeak-ng/espeak-ng/commit/059543ceb013ae9ed55212e0ee0376cf0023d250> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/issues/195> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/059543ceb013ae9ed55212e0ee0376cf0023d250>    Bug is at line 1791 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1791(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1556(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change conditional expression  Location: conditional expression |

| **Case 88** <https://github.com/espeak-ng/espeak-ng/commit/a621ad46e723860fad40f0a353a7c3dcb4db01ba> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/issues/155> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/espeak-ng/espeak-ng/commit/a621ad46e723860fad40f0a353a7c3dcb4db01ba>    Bug is at line 691 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  691(c\_dictionary\_25-cfg.dot) (IsLetterGroup)  …  ->661(c\_dictionary\_25-cfg.dot) (IsLetterGroup)  ->1779(c\_dictionary\_25-cfg.dot) (MatchRule)  …  ->1579(c\_dictionary\_25-cfg.dot) (MatchRule)  ->2288(c\_dictionary\_25-cfg.dot) (TranslateRules)  …  ->2200(c\_dictionary\_25-cfg.dot) (TranslateRules)  ->780(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  …  ->771(c\_dictionary\_25-cfg.dot) (Unpronouncable2)  ->848(c\_dictionary\_25-cfg.dot) (Unpronouncable)  …  ->787(c\_dictionary\_25-cfg.dot) (Unpronouncable)  ->278(c\_translateword\_26-cfg.dot) (TranslateWord3)  …  ->58(c\_translateword\_26-cfg.dot) (TranslateWord3)  ->150(c\_translate\_24-cfg.dot)(TranslateWord)  …  ->144(c\_translate\_24-cfg.dot)(TranslateWord)  ->404(c\_translate\_24-cfg.dot)(TranslateWord2)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateWord2)  ->1586(c\_translate\_24-cfg.dot)(TranslateClause)  …  ->1530(c\_translate\_24-cfg.dot)(TranslateClause)  ->1560(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  …  ->1530(c\_synthesize\_23-cfg.dot)(SpeakNextClause)  ->455(c\_speech\_22-cfg.dot)(Synthesize)  …  ->427(c\_speech\_22-cfg.dot)(Synthesize)  ->570(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  …  ->541(c\_speech\_22-cfg.dot)(sync\_espeak\_Synth)  ->678(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  …  ->663(c\_speech\_22-cfg.dot)(espeak\_ng\_Synthesize)  ->90(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  …  ->84(c\_espeak\_api\_21-cfg.dot)(espeak\_Synth)  ->313(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  …  ->305(c\_eSpeakService\_20-cfg.dot)(Java\_com\_reecedunn\_espeak\_SpeechSynthesis\_nativeSynthesize)  ->38(java\_SpeechSynthesis\_1-cfg.dot)(synthesize)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + return statement |

| **Case 89** <https://github.com/cossacklabs/themis/commit/8b877052f7c9384f52d81f6358882fe3a788cbd8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/issues/155> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/cossacklabs/themis/commit/8b877052f7c9384f52d81f6358882fe3a788cbd8>    Bug is at line 105 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  105(c\_soter\_sign\_ecdsa\_8-cfg.dot) (soter\_sign\_final\_ecdsa\_none\_pkcs8)  …  ->102(c\_soter\_sign\_ecdsa\_8-cfg.dot) (soter\_sign\_final\_ecdsa\_none\_pkcs8)  ->177(c\_soter\_sign\_4-cfg.dot) (soter\_sign\_final)  …  ->168(c\_soter\_sign\_4-cfg.dot) (soter\_sign\_final)  ->77(c\_secure\_message\_wrapper\_11-cfg.dot) (themis\_secure\_message\_signer\_proceed)  …  ->66(c\_secure\_message\_wrapper\_11-cfg.dot) (themis\_secure\_message\_signer\_proceed)  ->147(c\_secure\_message\_10-cfg.dot) (themis\_secure\_message\_sign)  …  ->131(c\_secure\_message\_10-cfg.dot) (themis\_secure\_message\_sign)  ->99(c\_themis\_message\_15-cfg.dot) (Java\_com\_cossacklabs\_themis\_SecureMessage\_process)  …  ->27(c\_themis\_message\_15-cfg.dot) (Java\_com\_cossacklabs\_themis\_SecureMessage\_process)  ->278(java\_SecureMessage\_26-cfg.dot) (wrap)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + return statement |

| **Case 90** <https://github.com/cossacklabs/themis/commit/8b877052f7c9384f52d81f6358882fe3a788cbd8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/espeak-ng/espeak-ng/issues/155> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/cossacklabs/themis/commit/8b877052f7c9384f52d81f6358882fe3a788cbd8>    Bug is at line 93 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  105(c\_soter\_sign\_ecdsa\_8-cfg.dot) (soter\_sign\_final\_ecdsa\_none\_pkcs8)  …  ->102(c\_soter\_sign\_ecdsa\_8-cfg.dot) (soter\_sign\_final\_ecdsa\_none\_pkcs8)  ->177(c\_soter\_sign\_4-cfg.dot) (soter\_sign\_final)  …  ->168(c\_soter\_sign\_4-cfg.dot) (soter\_sign\_final)  ->77(c\_secure\_message\_wrapper\_11-cfg.dot) (themis\_secure\_message\_signer\_proceed)  …  ->66(c\_secure\_message\_wrapper\_11-cfg.dot) (themis\_secure\_message\_signer\_proceed)  ->147(c\_secure\_message\_10-cfg.dot) (themis\_secure\_message\_sign)  …  ->131(c\_secure\_message\_10-cfg.dot) (themis\_secure\_message\_sign)  ->99(c\_themis\_message\_15-cfg.dot) (Java\_com\_cossacklabs\_themis\_SecureMessage\_process)  …  ->27(c\_themis\_message\_15-cfg.dot) (Java\_com\_cossacklabs\_themis\_SecureMessage\_process)  ->278(java\_SecureMessage\_26-cfg.dot) (wrap)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Null pointer reference |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + return statement |

| **Case 91**  <https://github.com/ninia/jep/commit/bd14a110911af80e7299767f43c570e2fca4d459> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/77> where it says    Thus, the symptom is compile failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/bd14a110911af80e7299767f43c570e2fca4d459>    Bug is at line 1332 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1332(c\_pyembed\_5-cfg.dot) (pyembed\_run\_pyc)  …  ->1318(c\_pyembed\_5-cfg.dot) (pyembed\_run\_pyc)  ->1295(c\_pyembed\_5-cfg.dot) (pyembed\_run)  …  ->1256(c\_pyembed\_5-cfg.dot) (pyembed\_run)  ->90(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_run)  …  ->84(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_run)  ->472(java\_jep\_20-cfg.dot) (runScript)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a function call  Location: function call |

| **Case 92**  <https://github.com/ninia/jep/commit/d14567567ac1281e978790b9170c981113de282f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/79> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/d14567567ac1281e978790b9170c981113de282f>    Bug is at line 264 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  264(c\_pyembed\_5-cfg.dot) (pyembed\_version\_unsafe)  …  ->255(c\_pyembed\_5-cfg.dot) (pyembed\_version\_unsafe)  ->229(c\_pyembed\_5-cfg.dot) (pyembed\_startup)  …  ->213(c\_pyembed\_5-cfg.dot) (pyembed\_startup)  ->65(c\_maininterpreter\_6-cfg.dot) (Java\_jep\_MainInterpreter\_initializePython)  …  ->62(c\_maininterpreter\_6-cfg.dot) (Java\_jep\_MainInterpreter\_initializePython)  ->149(java\_MainInterpreter\_21-cfg.dot) (run)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument in the function call  Location: function call + assignment statement |

| **Case 93**  <https://github.com/ninia/jep/commit/8144158fdaf3db44681e5eb4ef9a77907e4746b4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/40> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/8144158fdaf3db44681e5eb4ef9a77907e4746b4>    Bug is at line 1242 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1242(c\_pyembed\_5-cfg.dot) (pyembed\_box\_py)  …  ->1019(c\_pyembed\_5-cfg.dot) (pyembed\_box\_py)  ->776(c\_pyembed\_5-cfg.dot) (pyembed\_invoke)  …  ->719(c\_pyembed\_5-cfg.dot) (pyembed\_invoke)  ->708(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  …  ->679(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  ->112(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  …  ->100(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  ->394(java\_jep\_20-cfg.dot) (invoke)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location:conditional expression |

| **Case 94**  <https://github.com/ninia/jep/commit/3d33ea017acb4301e9576d8370b3305a424af80a> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/ninia/jep/issues/36where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/3d33ea017acb4301e9576d8370b3305a424af80a>    Bug is at line 978 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  978(c\_pyembed\_5-cfg.dot) (pyembed\_box\_py)  …  ->909(c\_pyembed\_5-cfg.dot) (pyembed\_box\_py)  ->685(c\_pyembed\_5-cfg.dot) (pyembed\_invoke)  …  ->633(c\_pyembed\_5-cfg.dot) (pyembed\_invoke)  ->622(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  …  ->595(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  ->112(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  …  ->100(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  ->394(java\_jep\_20-cfg.dot) (invoke)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change data type  Location: assignment statement |

| **Case 95**  <https://github.com/ninia/jep/commit/3d33ea017acb4301e9576d8370b3305a424af80a> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/ninia/jep/issues/36where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/3d33ea017acb4301e9576d8370b3305a424af80a>    Bug is at line 307 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  307(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_set\_\_JLjava\_lang\_String\_2J)  …  ->302(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_set\_\_JLjava\_lang\_String\_2J)  ->871(java\_jep\_20-cfg.dot) (set)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change data type of argument  Location: function call |

| **Case 96**  <https://github.com/ninia/jep/commit/767509de81f3a11b1ca54e23619d3226944a141e> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/ninia/jep/issues/36where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/767509de81f3a11b1ca54e23619d3226944a141e>    Bug is at line 229 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  229(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_close)  …  ->227(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_close)  ->1116(java\_jep\_20-cfg.dot) (close)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument  Location: function call |

| **Case 97**  <https://github.com/ninia/jep/commit/340788b3ae7a3e4ccfc610f73d43afeba508241c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/17> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/340788b3ae7a3e4ccfc610f73d43afeba508241c>    Bug is at line 337 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  337(c\_pyembed\_5-cfg.dot) (pyembed\_thread\_close)  …  ->312(c\_pyembed\_5-cfg.dot) (pyembed\_thread\_close)  ->194(c\_jep\_4-cfg.dot) (pyembed\_thread\_close)  …  ->191(c\_jep\_4-cfg.dot) (pyembed\_thread\_close)  ->1116(java\_jep\_20-cfg.dot) (close)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Reference count misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the foreign function API  Location: function call |

| **Case 98**  <https://github.com/ninia/jep/commit/778b516930bdc11ecbb7751d560d373964009639> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/17> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/778b516930bdc11ecbb7751d560d373964009639>    Bug is at line 356 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  356(c\_pyembed\_5-cfg.dot) (pyembed\_thread\_close)  …  ->312(c\_pyembed\_5-cfg.dot) (pyembed\_thread\_close)  ->194(c\_jep\_4-cfg.dot) (pyembed\_thread\_close)  …  ->191(c\_jep\_4-cfg.dot) (pyembed\_thread\_close)  ->1116(java\_jep\_20-cfg.dot) (close)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Reference count misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the foreign function API  Location: function call + if/branching statement |

| **Case 99**  <https://github.com/ninia/jep/commit/d2ab3f0343a518948ee44f09279e7ccf6922b856> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/17> where it says    Thus, the symptom is compatibility error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/d2ab3f0343a518948ee44f09279e7ccf6922b856>    Bug is at line 304 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  304(c\_pyjclass\_6-cfg.dot) (pyjclass\_add\_inner\_classes)  …  ->212(c\_pyjclass\_6-cfg.dot) (pyjclass\_add\_inner\_classes)  ->199(c\_pyjclass\_7-cfg.dot) (pyjclass\_init)  …  ->184(c\_pyjclass\_7-cfg.dot) (pyjclass\_init)  ->216(c\_pyjclass\_7-cfg.dot) (PyJClass\_Wrap)  …  ->212(c\_pyjclass\_7-cfg.dot) (PyJClass\_Wrap)  ->192(c\_convert\_j2p\_8-cfg.dot) (jobject\_As\_PyJObject)  …  ->183(c\_convert\_j2p\_8-cfg.dot) (jobject\_As\_PyJObject)  ->174(c\_convert\_j2p\_8-cfg.dot) (jnumber\_As\_PyObject)  …  ->121(c\_convert\_j2p\_8-cfg.dot) (jnumber\_As\_PyObject)  ->214(c\_convert\_j2p\_8-cfg.dot) (jobject\_As\_PyObject)  …  ->203(c\_convert\_j2p\_8-cfg.dot) (jobject\_As\_PyObject)  ->1017(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_as)  …  ->981(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_as)  ->1111(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method\_as)  …  ->1078(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method\_as)  ->1141(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  …  ->1135(c\_pyembed\_5-cfg.dot) (pyembed\_invoke\_method)  ->112(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  …  ->100(c\_jep\_4-cfg.dot) (Java\_jep\_Jep\_invoke)  ->394(java\_jep\_20-cfg.dot) (invoke)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the foreign function API  Location: foreign function + conditional expression |

| **Case 100**  <https://github.com/ninia/jep/commit/d2ab3f0343a518948ee44f09279e7ccf6922b856> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ninia/jep/issues/17> where it says    Thus, the symptom is compatibility error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ninia/jep/commit/d2ab3f0343a518948ee44f09279e7ccf6922b856>    Bug is at line 418 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  418(c\_util\_8-cfg.dot) (process\_py\_exception)  …  ->227(c\_util\_8-cfg.dot) (process\_py\_exception)  ->555(c\_pyembed\_5-cfg.dot) (pyembed\_shared\_import)  …  ->543(c\_pyembed\_5-cfg.dot) (pyembed\_shared\_import)  ->76(c\_maininterpreter\_6-cfg.dot) (Java\_jep\_MainInterpreter\_sharedImportInternal)  …  ->73(c\_maininterpreter\_6-cfg.dot) (Java\_jep\_MainInterpreter\_sharedImportInternal)  ->169(java\_MainInterpreter\_21-cfg.dot) (run)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | NewStringUTF | Foreign | String Operations | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the if statement and change the argument in foreign function  Location: foreign function + if/branching statement |

| **Case 101**  <https://github.com/termux/termux-x11/commit/c84625625f1dfaa93e2e404003571942f79ce4be> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/termux/termux-x11/issues/231> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/termux/termux-x11/commit/c84625625f1dfaa93e2e404003571942f79ce4be>    Bug is at line 65 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  65(c\_android\_3-cfg.dot) (Java\_com\_termux\_x11\_CmdEntryPoint\_start)  …  ->49(c\_android\_3-cfg.dot) (Java\_com\_termux\_x11\_CmdEntryPoint\_start)  ->169(java\_CmdEntryPoint\_20-cfg.dot) (CmdEntryPoint)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Config error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the if statement to check env  Location: if statement |

| **Case 102**  <https://github.com/uclouvain/openjpeg/commit/4841292b5df8f5ed3c92f1760769428ad7500b7a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/1> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/4841292b5df8f5ed3c92f1760769428ad7500b7a>    Bug is at line 1418 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1418(c\_convert\_2-cfg.dot) (imagetopgx)  …  ->1358(c\_convert\_2-cfg.dot) (imagetopgx)  ->738(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + memory manegement  Location: if/branching statement + function call |

| **Case 103**  <https://github.com/uclouvain/openjpeg/commit/15f081c89650dccee4aa4ae66f614c3fdb268767> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/726> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/15f081c89650dccee4aa4ae66f614c3fdb268767>    Bug is at line 970 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  970(c\_convertbmp\_3-cfg.dot) (imagetobmp)  …  ->785(c\_convertbmp\_3-cfg.dot) (imagetobmp)  ->747(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression  Location: conditional expression |

| **Case 104**  <https://github.com/uclouvain/openjpeg/commit/d48be27f5aa018d398cf9369b2c664bee4ce6e19> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/642> where it says    Thus, the symptom is compile failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/d48be27f5aa018d398cf9369b2c664bee4ce6e19>    Bug is at line 55 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  55(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_alloc\_n)  …  ->39(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_alloc\_n)  ->204(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_malloc)  …  ->202(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_malloc)  ->1135(c\_opj\_malloc\_4-cfg.dot) (opj\_image\_data\_alloc)  …  ->1133(c\_opj\_malloc\_4-cfg.dot) (opj\_image\_data\_alloc)  ->76(c\_image\_6-cfg.dot) (opj\_image\_create)  …  ->40(c\_image\_6-cfg.dot) (opj\_image\_create)  ->1811(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  …  ->1734(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  ->2040(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: assert statement |

| **Case 105**  <https://github.com/uclouvain/openjpeg/commit/05b3afd28f8c3f9551d551c565609062f57f5837> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/635> where it says    Thus, the symptom is compile failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/05b3afd28f8c3f9551d551c565609062f57f5837>    Bug is at line 9972 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  9972(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  …  ->9942(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  ->670(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1915(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + loop |

| **Case 106**  <https://github.com/uclouvain/openjpeg/commit/05b3afd28f8c3f9551d551c565609062f57f5837> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/635> where it says    Thus, the symptom is compile failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/05b3afd28f8c3f9551d551c565609062f57f5837>    Bug is at line 52 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  52(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_alloc\_n)  …  ->39(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_alloc\_n)  ->128(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_malloc)  …  ->126(c\_opj\_malloc\_4-cfg.dot) (opj\_aligned\_malloc)  ->1135(c\_opj\_malloc\_4-cfg.dot) (opj\_image\_data\_alloc)  …  ->1133(c\_opj\_malloc\_4-cfg.dot) (opj\_image\_data\_alloc)  ->1135(c\_openjpeg\_8-cfg.dot) (opj\_image\_data\_alloc)  …  ->1133(c\_openjpeg\_8-cfg.dot) (opj\_image\_data\_alloc)  ->76(c\_image\_6-cfg.dot) (opj\_image\_create)  …  ->40(c\_image\_6-cfg.dot) (opj\_image\_create)  ->1811(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  …  ->1734(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  ->2040(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 107**  <https://github.com/uclouvain/openjpeg/commit/83249c318fa9a133cb992973f84e15d3ba3182e0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/624> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/83249c318fa9a133cb992973f84e15d3ba3182e0>    Bug is at line 9972 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  9972(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  …  ->9942(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  ->670(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1915(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 108**  <https://github.com/uclouvain/openjpeg/commit/55dbf8acff9afab1591b6a094b744d8426a32dd4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/599> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/55dbf8acff9afab1591b6a094b744d8426a32dd4>    Bug is at line 8607 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  8607(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_cstr\_index)  …  ->8595(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_cstr\_index)  ->8572(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_decompress)  …  ->8539(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_decompress)  ->277(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a free function after use  Location: function call |

| **Case 109**  <https://github.com/uclouvain/openjpeg/commit/c16c91797f4b15eb55d56f85fd497d588331e71f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/598> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/c16c91797f4b15eb55d56f85fd497d588331e71f>    Bug is at line 10002 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  10002(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  …  ->9935(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  ->670(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1915(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a free function after use  Location: function call |

| **Case 110**  <https://github.com/uclouvain/openjpeg/commit/c16c91797f4b15eb55d56f85fd497d588331e71f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/598> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/c16c91797f4b15eb55d56f85fd497d588331e71f>    Bug is at line 10007 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  10007(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  …  ->9935(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  ->670(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1915(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a free function before return statement  Location: function call + if/branching statement |

| **Case 111**  <https://github.com/uclouvain/openjpeg/commit/5d953558de8dc4d939b8147fe9547533a8804297> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/597> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/5d953558de8dc4d939b8147fe9547533a8804297>    Bug is at line 9736 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  9736(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_one\_tile )  …  ->9634(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_one\_tile )  ->9761(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_one\_tile )  …  ->9755(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_one\_tile )  ->9882(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_tile )  …  ->9812(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_tile )  ->256(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a free function before return statement  Location: function call |

| **Case 112**  <https://github.com/uclouvain/openjpeg/commit/5d953558de8dc4d939b8147fe9547533a8804297> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/597> where it says    Thus, the symptom is no description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/5d953558de8dc4d939b8147fe9547533a8804297>    Bug is at line 5579 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  5579(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mco )  …  ->5545(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mco )  ->4873(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mct\_data\_group )  …  ->4832(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mct\_data\_group )  ->10380(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_writing )  …  ->10326(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_writing )  ->10073(c\_j2k\_8-cfg.dot) (opj\_j2k\_start\_compress )  …  ->10033(c\_j2k\_8-cfg.dot) (opj\_j2k\_start\_compress )  ->681(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: let the variable point to correct memory zone  Location: assignment statement |

| **Case 113**  [https://github.com/uclouvain/openjpeg/commit/0df90afdf743c85fe943041f5716ca072b4678d1](https://github.com/uclouvain/openjpeg/commit/0df90afdf743c85fe943041f5716ca072b4678d1#diff-c844731b17ba814529d31ed0060cda7c2a3fa0ef1363e0e226dceca07b3f3db3) |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/246> where it says    Thus, the symptom is redundant stdout/stderr |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/uclouvain/openjpeg/commit/0df90afdf743c85fe943041f5716ca072b4678d1](https://github.com/uclouvain/openjpeg/commit/0df90afdf743c85fe943041f5716ca072b4678d1#diff-c844731b17ba814529d31ed0060cda7c2a3fa0ef1363e0e226dceca07b3f3db3)    Bug is at line 1229 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1229(c\_jp2\_9-cfg.dot) (opj\_jp2\_apply\_cdef )  …  ->1213(c\_jp2\_9-cfg.dot) (opj\_jp2\_apply\_cdef )  ->1468(c\_jp2\_9-cfg.dot) (opj\_jp2\_decode )  …  ->1427(c\_jp2\_9-cfg.dot) (opj\_jp2\_decode )  ->300(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change used function call  Location: function call |

| **Case 114**  <https://github.com/uclouvain/openjpeg/commit/c3d9719cd4b48ab02ab984350f267bdfea5c6611> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/459> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/c3d9719cd4b48ab02ab984350f267bdfea5c6611>    Bug is at line 4906 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  4906(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mct\_data\_group )  …  ->4885(c\_j2k\_8-cfg.dot) (opj\_j2k\_write\_mct\_data\_group )  ->10449(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_writing )  …  ->10420(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_writing )  ->10190(c\_j2k\_8-cfg.dot) (opj\_j2k\_start\_compress )  …  ->10152(c\_j2k\_8-cfg.dot) (opj\_j2k\_start\_compress )  ->681(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the logic in loop and if statement  Location: loop statement + if/branching statement |

| **Case 115**  <https://github.com/uclouvain/openjpeg/commit/c423cc84e7be79051a7f9631fa26aa7d072361f2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/442> where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/c423cc84e7be79051a7f9631fa26aa7d072361f2>    Bug is at line 509 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  509(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  …  ->506(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  ->566(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the initialization value  Location: assignment statement |

| **Case 116**  <https://github.com/uclouvain/openjpeg/commit/b190a91ab5a147cfdf10dde57354bec90a1f9258> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/442> where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/b190a91ab5a147cfdf10dde57354bec90a1f9258>    Bug is at line 255 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  255(c\_opj\_decompress\_10-cfg.dot) (parse\_precision )  …  ->241(c\_opj\_decompress\_10-cfg.dot) (parse\_precision )  ->706(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  …  ->506(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  ->566(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type of variable  Location: variable definition statement |

| **Case 117**  <https://github.com/uclouvain/openjpeg/commit/b190a91ab5a147cfdf10dde57354bec90a1f9258> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/442> where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/b190a91ab5a147cfdf10dde57354bec90a1f9258>    Bug is at line 273 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  273(c\_opj\_decompress\_10-cfg.dot) (parse\_precision )  …  ->241(c\_opj\_decompress\_10-cfg.dot) (parse\_precision )  ->706(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  …  ->506(c\_opj\_decompress\_10-cfg.dot) (parse\_cmdline\_decoder )  ->566(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + type conversion  Location: conditional expression |

| **Case 118**  <https://github.com/uclouvain/openjpeg/commit/28c6f547987e8cbe5ccaef622da4cf6667068989> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/470> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/28c6f547987e8cbe5ccaef622da4cf6667068989>    Bug is at line 8085 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  8085(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_tile)  …  ->8019(c\_j2k\_8-cfg.dot) (opj\_j2k\_decode\_tile)  ->248(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + add a function call  Location: if/branching statement + function call |

| **Case 119**  <https://github.com/uclouvain/openjpeg/commit/775ee877761e3768f615b40729520b632c128645> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/235> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/775ee877761e3768f615b40729520b632c128645>    Bug is at line 916 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  916(c\_jp2\_9-cfg.dot) (opj\_jp2\_check\_color )  …  ->819(c\_jp2\_9-cfg.dot) (opj\_jp2\_check\_color )  ->1429(c\_jp2\_9-cfg.dot) (opj\_jp2\_decode )  …  ->1414(c\_jp2\_9-cfg.dot) (opj\_jp2\_decode )  ->300(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + data processing  Location: if/branching statement + loop |

| **Case 120**  <https://github.com/uclouvain/openjpeg/commit/775ee877761e3768f615b40729520b632c128645> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/447>, which points to an earlier report at <https://issues.chromium.org/issues/41145406> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/775ee877761e3768f615b40729520b632c128645>    Bug is at line 2038 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  2038(c\_jp2\_9-cfg.dot) (opj\_jp2\_read\_header\_procedure )  …  ->2002(c\_jp2\_9-cfg.dot) (opj\_jp2\_read\_header\_procedure )  ->1944(c\_jp2\_9-cfg.dot) (opj\_jp2\_setup\_end\_header\_reading )  …  ->1940(c\_jp2\_9-cfg.dot) (opj\_jp2\_setup\_end\_header\_reading )  ->1889(c\_jp2\_9-cfg.dot) (opj\_jp2\_end\_decompress )  …  ->1878(c\_jp2\_9-cfg.dot) (opj\_jp2\_end\_decompress )  ->305(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + data processing  Location: if/branching statement |

| **Case 121**  <https://github.com/uclouvain/openjpeg/commit/0b1f8eb6f72c2a6c741d896c7038b9fff4a2d02d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/pull/529> where it says    Thus, the symptom is no apparent symptom. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/0b1f8eb6f72c2a6c741d896c7038b9fff4a2d02d>    Bug is at line 10018 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  10018(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  …  ->9976(c\_j2k\_8-cfg.dot) (opj\_j2k\_encode)  ->672(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  …  ->657(c\_openjpeg\_8-cfg.dot) (opj\_create\_compress)  ->2058(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1915(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete a function call  Location: function call |

| **Case 122**  <https://github.com/uclouvain/openjpeg/commit/d1b053afe2916ad65e53d2c7f4d66e5a8d1df3e7> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/uclouvain/openjpeg/issues/470> , which points to an earlier report at <https://github.com/uclouvain/openjpeg/issues/392> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/d1b053afe2916ad65e53d2c7f4d66e5a8d1df3e7>      Bug is at line 7630 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  7630(c\_jp2\_9-cfg.dot) (opj\_j2k\_destroy )  …  ->7583(c\_jp2\_9-cfg.dot) (opj\_j2k\_destroy )  ->222(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 123**  <https://github.com/uclouvain/openjpeg/commit/dc43ebf96c079fb2a91a0e18f4d8f180855dd9ac> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/uclouvain/openjpeg/issues/470> , which points to an earlier report at <https://github.com/uclouvain/openjpeg/issues/392> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/dc43ebf96c079fb2a91a0e18f4d8f180855dd9ac>    Bug is at line 1383 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1383(c\_convert\_2-cfg.dot) (imagetopnm )  …  ->1353(c\_convert\_2-cfg.dot) (imagetopnm )  ->729(c\_JavaOpenJPEGDecoder\_5-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 124**  <https://github.com/uclouvain/openjpeg/commit/dc43ebf96c079fb2a91a0e18f4d8f180855dd9ac> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/uclouvain/openjpeg/issues/427> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/dc43ebf96c079fb2a91a0e18f4d8f180855dd9ac>    Bug is at line 2036 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  2036(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_siz)  …  ->1943(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_siz)  ->1293(c\_j2k\_8-cfg.dot) (j2k\_memory\_marker\_handler\_tab)  …  ->1284(c\_j2k\_8-cfg.dot) (j2k\_memory\_marker\_handler\_tab)  ->7514(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_marker\_handler)  …  ->7511(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_marker\_handler)  ->7229(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header\_procedure)  …  ->7185(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header\_procedure)  ->6884(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_reading)  …  ->6879(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_reading)  ->6854(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header)  …  ->6827(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header)  ->219(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete boundary check + add function call to assign value  Location: function call + assignment |

| **Case 125**  <https://github.com/uclouvain/openjpeg/commit/237ddd72f15fffe26e446604b2019604d9ae5211> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/uclouvain/openjpeg/issues/427> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/237ddd72f15fffe26e446604b2019604d9ae5211>    Bug is at line 4038 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  4038(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_sot )  …  ->4021(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_sot )  ->1286(c\_j2k\_8-cfg.dot) (j2k\_memory\_marker\_handler\_tab)  …  ->1284(c\_j2k\_8-cfg.dot) (j2k\_memory\_marker\_handler\_tab)  ->7514(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_marker\_handler)  …  ->7511(c\_j2k\_8-cfg.dot) (opj\_j2k\_get\_marker\_handler)  ->7229(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header\_procedure)  …  ->7185(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header\_procedure)  ->6884(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_reading)  …  ->6879(c\_j2k\_8-cfg.dot) (opj\_j2k\_setup\_header\_reading)  ->6854(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header)  …  ->6827(c\_j2k\_8-cfg.dot) (opj\_j2k\_read\_header)  ->219(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_JavaOpenJPEGDecoder\_5-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the conditional expression  Location: conditional expression |

| **Case 126**  <https://github.com/uclouvain/openjpeg/commit/55dbf8acff9afab1591b6a094b744d8426a32dd4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/599> where it says    Thus, the symptom is memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/55dbf8acff9afab1591b6a094b744d8426a32dd4>    Bug is at line 8644 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  8644(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_decompress)  …  ->8539(c\_j2k\_8-cfg.dot) (opj\_j2k\_create\_decompress)  ->277(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  …  ->184(c\_openjpeg\_8-cfg.dot) (opj\_create\_decompress)  ->624(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  …  ->496(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaDecoder\_internalDecodeJ2KtoImage)  ->127(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (decodeJ2KtoImage)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: Conditional Preprocessor Directives |

| **Case 127**  <https://github.com/uclouvain/openjpeg/commit/5855da5a156443f470a0c446e1740ab8aaf09230> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/635> where it says    Thus, the symptom is memory access violation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/5855da5a156443f470a0c446e1740ab8aaf09230>    Bug is at line 43 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  43(c\_image\_6-cfg.dot) (opj\_image\_create)  …  ->34(c\_image\_6-cfg.dot) (opj\_image\_create)  ->1811(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  …  ->1734(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  ->2040(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change used function  Location: function call + assignment |

| **Case 128**  <https://github.com/uclouvain/openjpeg/commit/5855da5a156443f470a0c446e1740ab8aaf09230> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/uclouvain/openjpeg/issues/635> where it says    Thus, the symptom is no description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/uclouvain/openjpeg/commit/5855da5a156443f470a0c446e1740ab8aaf09230>    Bug is at line 853 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  43(c\_image\_6-cfg.dot) (opj\_image\_create)  …  ->34(c\_image\_6-cfg.dot) (opj\_image\_create)  ->1811(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  …  ->1734(c\_JavaOpenJPEG\_7-cfg.dot) (loadImage)  ->2040(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  …  ->1914(c\_JavaOpenJPEG\_7-cfg.dot) (Java\_org\_openJpeg\_OpenJPEGJavaEncoder\_internalEncodeImageToJ2K)  ->165(java\_OpenJPEGJavaDecoder\_21-cfg.dot) (encodeImageToJ2K)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetMethodID | Foreign | Calling Instance Methods | | GetArrayLength | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change used function  Location: function call + assignment |

| **Case 129**  <https://github.com/TurboVNC/turbovnc/commit/34f383618eddd0dec95448c2dd1a82e54c19583b> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/pull/310> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/34f383618eddd0dec95448c2dd1a82e54c19583b>    Bug is at line 51 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  51(java\_Viewport\_11-cfg.dot) (Viewport)  …  ->98(java\_Viewport\_11-cfg.dot) (Viewport)  ->433(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  …  ->439(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  ->207(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_grabKeyboard)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a function call to init the value  Location: function call |

| **Case 130**  <https://github.com/TurboVNC/turbovnc/commit/ccb77c5111b2c1f7bf93a7f308794bc0dfa2dff4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/348> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/ccb77c5111b2c1f7bf93a7f308794bc0dfa2dff4>      Bug is at line 56 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  56(java\_Viewport\_11-cfg.dot) (Viewport)  …  ->77(java\_Viewport\_11-cfg.dot) (Viewport)  ->451(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  …  ->457(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  ->207(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_grabKeyboard)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: reset the input map  Location: function call + if/branching statement |

| **Case 131**  <https://github.com/TurboVNC/turbovnc/commit/6e8ceae35ce0a40998cf4005e12afaef2e8d6c92> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/190> where it says    Thus, the symptom is hang |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/6e8ceae35ce0a40998cf4005e12afaef2e8d6c92>    Bug is at line 337 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  337(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_setupExtInput)  ..  299(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_setupExtInput)  ->461(java\_Viewport\_11-cfg.dot) (setupExtInputHelper)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 132**  <https://github.com/TurboVNC/turbovnc/commit/04b26feefe7d17c427394e0479d48b79c603317a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/176> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/04b26feefe7d17c427394e0479d48b79c603317a>    Bug is at line 551 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  551(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_processExtInputEvent)  ..  477(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_processExtInputEvent)  ->601(java\_Viewport\_11-cfg.dot) (processExtInputEventHelper)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: return a value instead of assignment  Location: assignment statement |

| **Case 133**  <https://github.com/TurboVNC/turbovnc/commit/4baa42918508c1b77edba2e8675707046b8609d1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/121> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/4baa42918508c1b77edba2e8675707046b8609d1>    Bug is at line 253 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  253(java\_Viewport\_11-cfg.dot) (enableLionFS)  …  ->278(java\_Viewport\_11-cfg.dot) (enableLionFS)  ->78(java\_Viewport\_11-cfg.dot) (Viewport)  …  ->95(java\_Viewport\_11-cfg.dot) (Viewport)  ->407(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  …  ->414(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  ->207(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_grabKeyboard)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Config error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add if statement to check java version  Location: assignment statement |

| **Case 134**  <https://github.com/TurboVNC/turbovnc/commit/d063c2edd77a0923a8bc9a742d59dbe4eedacfab> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/105> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/d063c2edd77a0923a8bc9a742d59dbe4eedacfab>    Bug is at line 502 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  502(java\_DesktopWindow\_13-cfg.dot) (mousePressed)  ->484(java\_DesktopWindow\_13-cfg.dot) (mouseCB)  …  ->489(java\_DesktopWindow\_13-cfg.dot) (mouseCB)  ->643(java\_Viewport\_11-cfg.dot) (processExtInputEventHelper)  …  ->649(java\_Viewport\_11-cfg.dot) (processExtInputEventHelper)  ->466(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_processExtInputEvent)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + function call |

| **Case 135**  <https://github.com/TurboVNC/turbovnc/commit/4336781905bde2be234ad1f01dc041cb12579452> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/66> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/4336781905bde2be234ad1f01dc041cb12579452>    Bug is at line 1553 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  1553(java\_CConn\_14-cfg.dot) (getOptions)  …  ->1589(java\_CConn\_14-cfg.dot) (getOptions)  ->941(java\_CConn\_14-cfg.dot) (recreateViewport)  …  ->970(java\_CConn\_14-cfg.dot) (recreateViewport)  ->443(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  …  ->446(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  ->88(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_x11FullScreen)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | SetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change boundary check against value range/limit  Location: conditional expression |

| **Case 136**  <https://github.com/TurboVNC/turbovnc/commit/5103f12fa6e0f1cc8a8e844ed2bcdbe1dbcf9b0e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/209> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/5103f12fa6e0f1cc8a8e844ed2bcdbe1dbcf9b0e>    Bug is at line 1622 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  1622(java\_CConn\_14-cfg.dot) (getOptions)  …  ->1699(java\_CConn\_14-cfg.dot) (getOptions)  ->941(java\_CConn\_14-cfg.dot) (recreateViewport)  …  ->971(java\_CConn\_14-cfg.dot) (recreateViewport)  ->443(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  …  ->446(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  ->88(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_x11FullScreen)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | SetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 137**  <https://github.com/TurboVNC/turbovnc/commit/5103f12fa6e0f1cc8a8e844ed2bcdbe1dbcf9b0e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/209> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/5103f12fa6e0f1cc8a8e844ed2bcdbe1dbcf9b0e>    Bug is at line 1655 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  1655(java\_CConn\_14-cfg.dot) (getOptions)  …  ->1699(java\_CConn\_14-cfg.dot) (getOptions)  ->941(java\_CConn\_14-cfg.dot) (recreateViewport)  …  ->971(java\_CConn\_14-cfg.dot) (recreateViewport)  ->443(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  …  ->446(java\_Viewport\_11-cfg.dot) (x11FullScreenHelper)  ->88(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_x11FullScreen)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | SetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change function call to assignment  Location: function call + assignment |

| **Case 138**  <https://github.com/TurboVNC/turbovnc/commit/84692b335ef6eb1aef7a5c10b3764959f5f34a41> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/13> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/84692b335ef6eb1aef7a5c10b3764959f5f34a41>    Bug is at line 1201 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1201(java\_CConn\_14-cfg.dot) (sizeWindow)  …  ->1152(java\_CConn\_14-cfg.dot) (sizeWindow)  ->1235(java\_CConn\_14-cfg.dot) (reconfigureViewport)  …  ->1225(java\_CConn\_14-cfg.dot) (reconfigureViewport)  ->1589(java\_CConn\_14-cfg.dot) (getOptions)  …  ->1584(java\_CConn\_14-cfg.dot) (getOptions)  ->979(java\_CConn\_14-cfg.dot) (recreateViewport)  ->978(java\_CConn\_14-cfg.dot) (recreateViewport)  ->412(java\_Viewport\_11-cfg.dot) (setupExtInputHelper)  …  ->401(java\_Viewport\_11-cfg.dot) (setupExtInputHelper)  ->422(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_setupExtInput)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: function call + assignment statement |

| **Case 139**  <https://github.com/TurboVNC/turbovnc/commit/9d6a396b5ba8f2f3e8e2bf13eecbb47fea8ffbcf> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/28> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/9d6a396b5ba8f2f3e8e2bf13eecbb47fea8ffbcf>    Bug is at line 1551 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  1551(java\_CConn\_14-cfg.dot) (getOptions)  ->938(java\_CConn\_14-cfg.dot) (recreateViewport)  …  ->951(java\_CConn\_14-cfg.dot) (recreateViewport)  ->466(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  …  ->472(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  ->142(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_grabKeyboard)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + function call + assignment statement |

| **Case 140**  <https://github.com/TurboVNC/turbovnc/commit/f369b9a235daf3c9e061da71a3afcd1ad21a0f26> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/TurboVNC/turbovnc/issues/11> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/TurboVNC/turbovnc/commit/f369b9a235daf3c9e061da71a3afcd1ad21a0f26>    Bug is at line 938 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  938(java\_CConn\_14-cfg.dot) (getSpannedSize)  …  ->891(java\_CConn\_14-cfg.dot) (getSpannedSize)  ->1011(java\_CConn\_14-cfg.dot) (sizeWindow)  …  ->1005(java\_CConn\_14-cfg.dot) (sizeWindow)  ->1068(java\_CConn\_14-cfg.dot) (reconfigureViewport)  …  ->1058(java\_CConn\_14-cfg.dot) (reconfigureViewport)  ->1407(java\_CConn\_14-cfg.dot) (getOptions)  …  ->1402(java\_CConn\_14-cfg.dot) (getOptions)  ->888(java\_CConn\_14-cfg.dot) (recreateViewport)  …  ->886(java\_CConn\_14-cfg.dot) (recreateViewport)  ->487(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  …  ->472(java\_Viewport\_11-cfg.dot) (grabKeyboardHelper)  ->212(c\_turbovnchelper\_5-cfg.dot) (Java\_com\_turbovnc\_vncviewer\_Viewport\_grabKeyboard)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetObjectClass | Foreign | Object Operations | | GetLongField | Foreign | Accessing Fields of Objects | | GetIntField | Foreign | Accessing Fields of Objects | | GetFieldID | Foreign | Accessing Fields of Objects | | FindClass | Foreign | Class Operations | | GetObjectField | Foreign | Accessing Fields of Objects | | SetIntArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + function call + assignment statement |

| **Case 141**  <https://github.com/switch-iot/hin2n/commit/e1ce97ce0a1250ed319faf7d8d17dd3af11f7a20> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/switch-iot/hin2n/issues/27> where it says    Thus, the symptom is hang |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/switch-iot/hin2n/commit/e1ce97ce0a1250ed319faf7d8d17dd3af11f7a20>    Bug is at line 125 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  125(java\_N2NService\_11-cfg.dot) (stop)  ->96(c\_edge\_jni\_3-cfg.dot) (Java\_wang\_switchy\_hin2n\_service\_N2NService\_stopEdge)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a thread to execute function  Location: function call |

| **Case 142**  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/eclipse/kura/pull/1034where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0>    Bug is at line 144 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  144(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  …  ->29(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  ->88(c\_LinuxUdevNative\_4-cfg.dot) (LinuxUdevNative)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | GetMethodID | Foreign | Calling Instance Methods | | NewStringUTF | Foreign | String Operations | | FindClass | Foreign | Class Operations | | NewObject | Foreign | Object Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Memory management error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add some function call to manage the memory  Location: function call + foreign function call |

| **Case 144**  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/eclipse/kura/pull/1034where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0>    Bug is at line 225 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  225(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  …  ->29(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  ->88(c\_LinuxUdevNative\_4-cfg.dot) (LinuxUdevNative)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | GetMethodID | Foreign | Calling Instance Methods | | NewStringUTF | Foreign | String Operations | | FindClass | Foreign | Class Operations | | NewObject | Foreign | Object Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change variable to const  Location: assignment |

| **Case 145**  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/eclipse/kura/pull/1034where it says    Thus, the symptom is warning message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/eclipse/kura/commit/af665d2ee682001c118eb11ac9c95af40a8bedd0>    Bug is at line 245 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  245(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  …  ->29(c\_LinuxUdev\_13-cfg.dot) (Java\_org\_eclipse\_kura\_linux\_usb\_LinuxUdevNative\_getUsbDevices)  ->88(c\_LinuxUdevNative\_4-cfg.dot) (LinuxUdevNative)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | GetMethodID | Foreign | Calling Instance Methods | | NewStringUTF | Foreign | String Operations | | FindClass | Foreign | Class Operations | | NewObject | Foreign | Object Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change variable to const  Location: assignment |

| **Case 146**  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2>    Bug is at line 60 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  60(c\_accept\_7-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_accept)  …  ->32(c\_accept\_7-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_accept)  ->148(java\_NativeUnixSocket\_13-cfg.dot) (accept)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: function call |

| **Case 147**  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2>    Bug is at line 300 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  300(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->32(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->143(java\_NativeUnixSocket\_13-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: function call + if/branching statement |

| **Case 148**  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/0c4bb3a395a2da561523a93faed7f008199da1f2>    Bug is at line 59 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  59(c\_connect\_9-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_connect)  …  ->32(c\_connect\_9-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_connect)  ->151(java\_NativeUnixSocket\_13-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change called function  Location: function call |

| **Case 149**  <https://github.com/kohlschutter/junixsocket/commit/27aa8d06169c95dfa3b1c603fc20ca4c6927296e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/27aa8d06169c95dfa3b1c603fc20ca4c6927296e>    Bug is at line 92 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  59(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->32(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->143(java\_NativeUnixSocket\_13-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add code chunk to close the serversocket  Location: loop + if/branching statement + assignment + function call |

| **Case 150**  <https://github.com/kohlschutter/junixsocket/commit/f8f280c8b27c47b17d627eb0efd2e33317d4b46d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/116> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/f8f280c8b27c47b17d627eb0efd2e33317d4b46d>    Bug is at line 398 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  398(c\_address\_6-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bytesToSockAddr)  …  ->354(c\_address\_6-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bytesToSockAddr)  ->143(java\_AFDatagramSocketImpl-11-cfg.dot) (send)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: type conversion  Location: type-conversion expression assignment |

| **Case 151**  <https://github.com/kohlschutter/junixsocket/commit/f8f280c8b27c47b17d627eb0efd2e33317d4b46d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/116> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/f8f280c8b27c47b17d627eb0efd2e33317d4b46d>    Bug is at line 402 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  402(c\_address\_6-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bytesToSockAddr)  …  ->354(c\_address\_6-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bytesToSockAddr)  ->143(java\_AFDatagramSocketImpl-11-cfg.dot) (send)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + assignment statement + function call |

| **Case 152**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/97> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 53 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  53(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  …  ->44(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  ->92(java\_AFSocketImpl\_12-cfg.dot) (createSocket)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + assignment statement + function call |

| **Case 153**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/97> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 59,61 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  59(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  …  ->44(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  ->92(java\_AFSocketImpl\_12-cfg.dot) (createSocket)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Argument error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument in the function call  Location: assignment statement + function call |

| **Case 154**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/97> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 71 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  71(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  …  ->44(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_createSocket)  ->92(java\_AFSocketImpl\_12-cfg.dot) (createSocket)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 155**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/97> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 133 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  133(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_socketStatus)  …  ->95(c\_socket\_10-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_socketStatus)  ->84(java\_AFSocketCore-13-cfg.dot) (isConnected)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 156**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/103> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 39 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  39(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->31(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->319(java\_AFSocketImpl\_12-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the value source of the variable  Location: function call + foreign function |

| **Case 157**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/103> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 59 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  59(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->31(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->319(java\_AFSocketImpl\_12-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Argument error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument in function call  Location: function call + assignment |

| **Case 158**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/103> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 70 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  70(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->31(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->319(java\_AFSocketImpl\_12-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the value in initialization  Location: assignment |

| **Case 159**  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/103> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/2f8f096b84fba1ae26058d41cc03df47719379df>    Bug is at line 198 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  198(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->31(c\_bind\_8-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->319(java\_AFSocketImpl\_12-cfg.dot) (bind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | NIO Support |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 160**  <https://github.com/kohlschutter/junixsocket/commit/34af0c1be2539ea4a1390af711ff933463b834f9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/90> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/34af0c1be2539ea4a1390af711ff933463b834f9>    Bug is at line 198 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  198(c\_send\_20-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_write)  …  ->133(c\_send\_20-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_write)  ->408(java\_AFUNIXSocketImpl\_16-cfg.dot) (write)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + add branching to if statement  Location: if/branching statement |

| **Case 161**  <https://github.com/kohlschutter/junixsocket/commit/ba3d20f60e7fbf1a49ae1f4bac09111299d104ee> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/90> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/ba3d20f60e7fbf1a49ae1f4bac09111299d104ee>    Bug is at line 205 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  205(c\_receive\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_read)  …  ->175(c\_receive\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_read)  ->347(java\_AFUNIXSocketImpl\_16-cfg.dot) (read)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + error handling function call  Location: if/branching statement + function call |

| **Case 162**  <https://github.com/kohlschutter/junixsocket/commit/ba3d20f60e7fbf1a49ae1f4bac09111299d104ee> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/90> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/ba3d20f60e7fbf1a49ae1f4bac09111299d104ee>    Bug is at line 262 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  262(c\_receive\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_receive)  …  ->245(c\_receive\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_receive)  ->127(java\_AFUNIXDatagramSocketImpl\_17-cfg.dot) (recv)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + error handling function call  Location: if/branching statement + function call |

| **Case 163**  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/81> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079>      Bug is at line 689 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  689(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_accept)  …  ->655(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_accept)  ->136(java\_AFUNIXSocketImpl\_16-cfg.dot) (accept)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayElements | Foreign | Array Operations | | ReleaseByteArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 164**  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/81> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079>      Bug is at line 777 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  777(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  …  ->745(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_bind)  ->178(java\_AFUNIXSocketImpl\_16-cfg.dot) (accept)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayElements | Foreign | Array Operations | | ReleaseByteArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 165**  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/81> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/77a4f0c8cd641fc5dfcb62922981c32393c8e079>      Bug is at line 1108 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  1108(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_connect)  …  ->1069(c\_org\_newsclub\_net\_unix\_NativeUnixSocket\_18-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_connect)  ->263(java\_AFUNIXSocketImpl\_16-cfg.dot) (connect)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayElements | Foreign | Array Operations | | ReleaseByteArrayElements | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement |

| **Case 166**  <https://github.com/kohlschutter/junixsocket/commit/efacb5ae8e143a08bbd4c17e9a2909c64699c0bb> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/145> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/efacb5ae8e143a08bbd4c17e9a2909c64699c0bb>    Bug is at line 202 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  202(java\_AFSelector\_31-cfg.dot) (setOpsReady)  …  ->196(java\_AFSelector\_31-cfg.dot) (setOpsReady)  ->146(java\_AFSelector\_31-cfg.dot) (select0)  …  ->125(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + change used function call  Location: if/branching statement + function call |

| **Case 167**  <https://github.com/kohlschutter/junixsocket/commit/470e4260354d9eb016e6608bd4c84340749a9f5f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/145> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/470e4260354d9eb016e6608bd4c84340749a9f5f>    Bug is at line 188 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  188(java\_AFSelector\_31-cfg.dot) (setOpsReady)  …  ->181(java\_AFSelector\_31-cfg.dot) (setOpsReady)  ->141(java\_AFSelector\_31-cfg.dot) (select0)  …  ->119(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change used function call  Location: function call |

| **Case 168**  <https://github.com/kohlschutter/junixsocket/commit/889ca992f375f824797cbd577cbb0b162ccafb32> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/142> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/889ca992f375f824797cbd577cbb0b162ccafb32>    Bug is at line 200 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  200(java\_AFSelector\_31-cfg.dot) (initPollFd)  …  ->256(java\_AFSelector\_31-cfg.dot) (initPollFd)  ->113(java\_AFSelector\_31-cfg.dot) (select0)  …  ->119(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: function call |

| **Case 169**  <https://github.com/kohlschutter/junixsocket/commit/5e8a6e6fbad0be226cffb18961e1b4e6c6e88342> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/142> where it says    Thus, the symptom is error message |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/5e8a6e6fbad0be226cffb18961e1b4e6c6e88342>    Bug is at line 188 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  188(java\_AFSelector\_31-cfg.dot) (setOpsReady)  …  ->181(java\_AFSelector\_31-cfg.dot) (setOpsReady)  ->141(java\_AFSelector\_31-cfg.dot) (select0)  …  ->119(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + change used function call  Location: conditional expression + function call |

| **Case 170**  <https://github.com/kohlschutter/junixsocket/commit/09548dadf52ba6b4f939d0260c391503f478a6a5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/140> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/09548dadf52ba6b4f939d0260c391503f478a6a5>    Bug is at line 197 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  197(java\_NativeLibraryLoader\_32-cfg.dot) (load)  …  ->169(java\_NativeLibraryLoader\_32-cfg.dot) (load)  ->333(java\_NativeLibraryLoader\_32-cfg.dot) (loadLibrary)  …  ->306(java\_NativeLibraryLoader\_32-cfg.dot) (loadLibrary)  ->196(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_noop)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Compatibility error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete boundary check against value range/limit + delete function call  Location: if/branching statement + function call |

| **Case 171**  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/138> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235>    Bug is at line 233 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  233(java\_AFSelector\_31-cfg.dot) (initPollFd)  …  ->250(java\_AFSelector\_31-cfg.dot) (initPollFd)  ->114(java\_AFSelector\_31-cfg.dot) (select0)  …  ->120(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add loop to check value  Location: if/branching statement + function call + loop |

| **Case 172**  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/138> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235>    Bug is at line 243 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  243(java\_AFSelector\_31-cfg.dot) (initPollFd)  …  ->250(java\_AFSelector\_31-cfg.dot) (initPollFd)  ->114(java\_AFSelector\_31-cfg.dot) (select0)  …  ->120(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + function call |

| **Case 173**  <https://github.com/kohlschutter/junixsocket/commit/861daaacb9d433f3de4ee3118589a9eaebc0e4f8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/138> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/861daaacb9d433f3de4ee3118589a9eaebc0e4f8>    Bug is at line 195 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  195(java\_AFSelector\_31-cfg.dot) (setOpsReady)  …  ->184(java\_AFSelector\_31-cfg.dot) (setOpsReady)  ->142(java\_AFSelector\_31-cfg.dot) (select0)  …  ->107(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy:Do not create a lot of objects when working with AFSelector  Location: if/branching statement + function call |

| **Case 174**  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/138> where it says    Thus, the symptom is performance degradation |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/a7c310662393e4ce1a2c288a18a8bdbdb7140235>    Bug is at line 207 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  207(java\_AFSelector\_31-cfg.dot) (initPollFd)  …  ->211(java\_AFSelector\_31-cfg.dot) (initPollFd)  ->113(java\_AFSelector\_31-cfg.dot) (select0)  …  ->121(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: remove redundant data  Location: if/branching statement + function call |

| **Case 175**  <https://github.com/kohlschutter/junixsocket/commit/3994501308902ffb5d2eb7bce44971e3438b2a91> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/135> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/3994501308902ffb5d2eb7bce44971e3438b2a91>    Bug is at line 141 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  141(java\_AFCore\_32-cfg.dot) (read)  …  ->134(java\_AFCore\_32-cfg.dot) (read)  ->304(c\_receive\_20-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_receive)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a loop to check and loop the function call as long as both buffers have something to send/receive.  Location: loop + function call |

| **Case 176**  <https://github.com/kohlschutter/junixsocket/commit/3994501308902ffb5d2eb7bce44971e3438b2a91> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/135> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/3994501308902ffb5d2eb7bce44971e3438b2a91>    Bug is at line 196 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  196(java\_AFCore\_32-cfg.dot) (write)  …  ->204(java\_AFCore\_32-cfg.dot) (write)  ->210(c\_send\_22-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_write)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a loop to check and loop the function call as long as both buffers have something to send/receive.  Location: loop + function call |

| **Case 177**  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03>    Bug is at line 281 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  281(java\_AFSocketImpl\_12-cfg.dot) (accept0)  …  ->274(java\_AFSocketImpl\_12-cfg.dot) (accept0)  ->264(c\_filedescriptors\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_close)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | Monitor Operations | | MonitorExit | Foreign | Monitor Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add handling for the exception.  Location: if/branching statement |

| **Case 178**  <https://github.com/kohlschutter/junixsocket/commit/e10cf8fa679a2096ffefda84c8a8ea1033092885> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/e10cf8fa679a2096ffefda84c8a8ea1033092885>    Bug is at line 280 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  280(java\_AFServerSocket\_33-cfg.dot) (isBound)  ->279(java\_AFServerSocket\_33-cfg.dot) (isBound)  ->239(java\_AFServerSocket\_33-cfg.dot) (bind)  …  ->231(java\_AFServerSocket\_33-cfg.dot) (bind)  ->149(java\_AFServerSocket\_33-cfg.dot) (newInstance)  …  ->138(java\_AFServerSocket\_33-cfg.dot) (newInstance)  ->172(c\_socket\_21-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_socketStatus)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: check for a valid file descriptor  Location: return + function call |

| **Case 179**  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03>    Bug is at line 300 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  300(java\_AFServerSocket\_33-cfg.dot) (accept1)  …  ->297(java\_AFServerSocket\_33-cfg.dot) (accept1)  ->255(java\_AFSocketImpl\_12-cfg.dot) (accept0)  …  ->253(java\_AFSocketImpl\_12-cfg.dot) (accept0)  ->76(c\_accept\_7-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_accept)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | Monitor Operations |   Data/control flow direction:   * C->Java |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: throw a different type of exception object  Location: if/branching statement |

| **Case 180**  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03>    Bug is at line 588 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  588(java\_AFSocketImpl\_12-cfg.dot) (write)  …  ->595(java\_AFSocketImpl\_12-cfg.dot) (write)  ->210(c\_accept\_7-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_write)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Array Operations | | GetByteArrayRegion | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case hadnling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete boundary checking to allow zero-length writes to probe if the remote peer is alive  Location: if/branching statement |

| **Case 181**  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/121> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/42d30530ce33875fcfadd48e7b3e3fd623d99a03>    Bug is at line 483 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  483(java\_AFSocketImpl\_12-cfg.dot) (read)  …  ->497(java\_AFSocketImpl\_12-cfg.dot) (read)  ->215(c\_receive\_20-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_read)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetDirectBufferAddress | Foreign | Monitor Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Exception handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: throw a different type of exception object  Location: if/branching statement |

| **Case 182**  <https://github.com/kohlschutter/junixsocket/commit/34b5a5fba08c1cdb35af7879fa4c06f51dc122e0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/118> where it says    Thus, the symptom is hang |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/34b5a5fba08c1cdb35af7879fa4c06f51dc122e0>    Bug is at line 214 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  214(java\_AFCore\_32-cfg.dot) (getThreadLocalDirectByteBuffer)  …  ->229(java\_AFCore\_32-cfg.dot) (getThreadLocalDirectByteBuffer)  ->123(java\_AFCore\_32-cfg.dot) (read)  …  ->133(java\_AFCore\_32-cfg.dot) (read)  ->304(c\_receive\_20-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_receive)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete the maximum limit of variable  Location: if/branching statement + return statement |

| **Case 183**  <https://github.com/kohlschutter/junixsocket/commit/38e30ccc626f97a79df30b45b3ef45eee14e880f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/kohlschutter/junixsocket/issues/108> where it says    Thus, the symptom is incorect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/kohlschutter/junixsocket/commit/38e30ccc626f97a79df30b45b3ef45eee14e880f>    Bug is at line 216 |
| **Bug manifestation:**  From the bug location, forward search until origin in the other language:  Path:  216(java\_AFSelector\_31-cfg.dot) (setOpsReady)  ->121(java\_AFSelector\_31-cfg.dot) (select0)  …  ->127(java\_AFSelector\_31-cfg.dot) (select0)  ->345(c\_polling\_19-cfg.dot) (Java\_org\_newsclub\_net\_unix\_NativeUnixSocket\_poll)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetArrayLength | Foreign | Accessing Fields of Objects | | GetObjectField | Foreign | Array Operations | | GetIntArrayRegion | Foreign | Array Operations | | GetObjectArrayElement | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: updating the interestOps data structure in PollFd when a SelectionKey's interestOps were changed after the fact  Location: if/branching statement + return statement + function call + loop |

| **Case 184**  <https://github.com/Pi4J/pi4j-v2/commit/ec7b2049c310b35353b4d820bdf611885fcea512> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/Pi4J/pi4j-v2/issues/15> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/Pi4J/pi4j-v2/commit/ec7b2049c310b35353b4d820bdf611885fcea512>    Bug is at line 158 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  158(c\_com\_pi4j\_library\_pigpio\_internal\_PIGPIO\_12-cfg.dot) (Java\_com\_pi4j\_library\_pigpio\_internal\_PIGPIO\_gpioInitialise)  …  ->155(c\_com\_pi4j\_library\_pigpio\_internal\_PIGPIO\_12-cfg.dot) (Java\_com\_pi4j\_library\_pigpio\_internal\_PIGPIO\_gpioInitialise)  ->100(java\_PiGpioNativeImpl\_22-cfg.dot) (gpioInitialise)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: disable the signal handlers inside the PIGPIO library  Location: function call |

| **Case 185**  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90) |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ibmruntimes/openj9-openjdk-jdk8/pull/213> where it says    Thus, the symptom is Handshake failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90)      Bug is at line 431 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  431(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_CBCFinalEncrypt)  …  ->371(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_CBCFinalEncrypt)  ->272(java\_NativeCipherBlockChaining\_12-cfg.dot) (gpioInitialise)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetPrimitiveArrayCritical | Foreign | Array Operations | | ReleasePrimitiveArrayCritical | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Assign an initial value to the variable  Location: assignment statement |

| **Case 186**  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90) |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ibmruntimes/openj9-openjdk-jdk8/pull/213> where it says    Thus, the symptom is Handshake failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90)      Bug is at line 427 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  427(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_CBCFinalEncrypt)  …  ->371(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_CBCFinalEncrypt)  ->272(java\_NativeCipherBlockChaining\_12-cfg.dot) (gpioInitialise)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetPrimitiveArrayCritical | Foreign | Array Operations | | ReleasePrimitiveArrayCritical | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add switch to handle different cases  Location: if/branching statement + function call |

| **Case 187**  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90) |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ibmruntimes/openj9-openjdk-jdk8/pull/213> where it says    Thus, the symptom is Handshake failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90)      Bug is at line 540 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  540(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_GCMDecrypt)  …  ->478(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_GCMDecrypt)  ->542(java\_NativeGaloisCounterMode\_13-cfg.dot) (decryptFinal)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetPrimitiveArrayCritical | Foreign | Array Operations | | ReleasePrimitiveArrayCritical | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Assign an initial value to the variable  Location: assignment statement |

| **Case 188**  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90) |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ibmruntimes/openj9-openjdk-jdk8/pull/213> where it says    Thus, the symptom is Handshake failure |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/ibmruntimes/openj9-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90](https://github.com/ibmruntimes/semeru-openjdk-jdk8/commit/8ef74afc7077116cc21c6d670386cd61d5297e90)    Bug is at line 535 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  535(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_GCMDecrypt)  …  ->478(c\_NativeCrypto\_4-cfg.dot) (Java\_jdk\_crypto\_jniprovider\_NativeCrypto\_GCMDecrypt)  ->542(java\_NativeGaloisCounterMode\_13-cfg.dot) (decryptFinal)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetPrimitiveArrayCritical | Foreign | Array Operations | | ReleasePrimitiveArrayCritical | Foreign | Array Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add switch to handle different cases  Location: if/branching statement + function call |

| **Case 189**  <https://github.com/wolfSSL/wolfssljni/commit/77e91c1f46be5270bc1e47bae73fd284bd57d43f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/wolfSSL/wolfssljni/pull/79> where it says    Thus, the symptom is no description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/wolfSSL/wolfssljni/commit/77e91c1f46be5270bc1e47bae73fd284bd57d43f>    Bug is at line 180 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  Path:  180(c\_com\_wolfssl\_WolfSSL\_7-cfg.dot) (Java\_com\_wolfssl\_WolfSSL\_nativeFree)  …  ->174(c\_com\_wolfssl\_WolfSSL\_7-cfg.dot) (Java\_com\_wolfssl\_WolfSSL\_nativeFree)  ->110(java\_WolfSSLTest\_16-cfg.dot) (tstMethod)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the date type  Location: if/branching statement + function call |

| **Case 190**  <https://github.com/wolfSSL/wolfssljni/commit/77e91c1f46be5270bc1e47bae73fd284bd57d43f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/wolfSSL/wolfssljni/pull/79> where it says    Thus, the symptom is no description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/wolfSSL/wolfssljni/commit/77e91c1f46be5270bc1e47bae73fd284bd57d43f>    Bug is at line 400 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  400(c\_com\_wolfssl\_WolfSSL\_7-cfg.dot) (Java\_com\_wolfssl\_WolfSSL\_SSLv3\_1ServerMethod)  …  ->393(c\_com\_wolfssl\_WolfSSL\_7-cfg.dot) (Java\_com\_wolfssl\_WolfSSL\_SSLv3\_1ServerMethod)  ->230(java\_Server\_19-cfg.dot) (run)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the date type  Location: if/branching statement + function call |

| **Case 191**  <https://github.com/MDSplus/mdsplus/commit/471ff8ce79ad3f1c4872f291c77f9675c089f8c8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MDSplus/mdsplus/issues/1680> where it says    Thus, the symptom is mechanism does not work |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/MDSplus/mdsplus/commit/471ff8ce79ad3f1c4872f291c77f9675c089f8c8>    Bug is at line 81 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  81(c\_TreeGetSetShotId\_5-cfg.dot) (GetFileName)  …  ->69(c\_TreeGetSetShotId\_5-cfg.dot) (GetFileName)  ->117(c\_TreeGetSetShotId\_5-cfg.dot) (CreateShotIdFile)  …  ->112(c\_TreeGetSetShotId\_5-cfg.dot) (CreateShotIdFile)  ->132(c\_TreeGetSetShotId\_5-cfg.dot) (OpenShotIdFile)  …  ->122(c\_TreeGetSetShotId\_5-cfg.dot) (OpenShotIdFile)  ->147(c\_TreeGetSetShotId\_5-cfg.dot) (TreeGetCurrentShotId)  …  ->136(c\_TreeGetSetShotId\_5-cfg.dot) (TreeGetCurrentShotId)  ->2010(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_Tree\_getCurrent)  …  ->2001(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_Tree\_getCurrent)  ->343(java\_jDispatcherlp\_16-cfg.dot) (getCurrentShot)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the function to get the path  Location: if/branching statement + function call |

| **Case 192**  <https://github.com/MDSplus/mdsplus/commit/471ff8ce79ad3f1c4872f291c77f9675c089f8c8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MDSplus/mdsplus/issues/1680> where it says    Thus, the symptom is mechanism does not work |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/MDSplus/mdsplus/commit/471ff8ce79ad3f1c4872f291c77f9675c089f8c8>    Bug is at line 144 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  144(c\_TreeGetSetShotId\_5-cfg.dot) (TreeGetCurrentShotId)  …  ->136(c\_TreeGetSetShotId\_5-cfg.dot) (TreeGetCurrentShotId)  ->2010(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_Tree\_getCurrent)  …  ->2001(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_Tree\_getCurrent)  ->343(java\_jDispatcherlp\_16-cfg.dot) (getCurrentShot)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Changed the variable's initialization from dynamic allocation to static allocation.  Location: assignment |

| **Case 193**  <https://github.com/MDSplus/mdsplus/commit/08b765dbcc389317fea4c862ff7b1f0d60361bcf> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MDSplus/mdsplus/issues/1680> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/MDSplus/mdsplus/commit/08b765dbcc389317fea4c862ff7b1f0d60361bcf>    Bug is at line 144 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  1449(c\_TreeSegments\_6-cfg.dot) (get\_segment\_limits)  …  ->1447(c\_TreeSegments\_6-cfg.dot) (get\_segment\_limits)  ->2150(c\_TreeSegments\_6-cfg.dot) (\_TreeGetSegmentLimits  …  ->2147(c\_TreeSegments\_6-cfg.dot) (\_TreeGetSegmentLimits)  ->2583(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_getSegmentStart)  …  ->2575(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_getSegmentStart)  ->131(java\_MdsTreeNodeTest\_17-cfg.dot) (testArray)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Initialization error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the variable in struct initialization  Location: assignment |

| **Case 194**  <https://github.com/MDSplus/mdsplus/commit/687b2621aa4dcebc3cc5ee9cd780f38fefd8af87> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MDSplus/mdsplus/issues/1423> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/MDSplus/mdsplus/commit/687b2621aa4dcebc3cc5ee9cd780f38fefd8af87>    Bug is at line 151 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  151(c\_TreeFindNode\_8-cfg.dot) (\_TreeFindNode)  …  ->137(c\_TreeFindNode\_8-cfg.dot) (\_TreeFindNode)  ->634(c\_TreeAddNode\_7-cfg.dot) (\_TreeAddConglom)  …  ->610(c\_TreeAddNode\_7-cfg.dot) (\_TreeAddConglom)  ->64(c\_TreeAddNode\_7-cfg.dot) (TreeAddConglom)  …  ->62(c\_TreeAddNode\_7-cfg.dot) (TreeAddConglom)  ->3296(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_addDevice)  …  ->3281(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_addDevice)  ->470(java\_Tree\_18-cfg.dot) (addDevice)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | GetStringUTFChars | Foreign | String Operations | | ReleaseStringUTFChars | Foreign | String Operations |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + assign a value to variable  Location: assignment statement + if/branching statement |

| **Case 195**  <https://github.com/eclipse-windowbuilder/windowbuilder/commit/5cb095ed6e9a6ff8dd84e6f95bc566746c23520a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/eclipse-windowbuilder/windowbuilder/issues/609> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/eclipse-windowbuilder/windowbuilder/commit/5cb095ed6e9a6ff8dd84e6f95bc566746c23520a>      Bug is at line 260 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  260(c\_rcp\_11-cfg.dot) (Java\_org\_eclipse\_wb\_internal\_os\_macosx\_OSSupportMacOSXCarbon\_\_1fetchMenuVisualData)  …  ->256(c\_rcp\_11-cfg.dot) (Java\_org\_eclipse\_wb\_internal\_os\_macosx\_OSSupportMacOSXCarbon\_\_1fetchMenuVisualData)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the whole jni function  Location: assignment + function call + return |

| **Case 196**  <https://github.com/eclipse-platform/eclipse.platform.swt/commit/417af5e1c38e0df8bda5fd767a18153e529aa4a5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/eclipse-platform/eclipse.platform.swt/issues/462> where it says    Thus, the symptom is crash |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/eclipse-platform/eclipse.platform.swt/commit/417af5e1c38e0df8bda5fd767a18153e529aa4a5>      Bug is at line 12359 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  12359(c\_os\_12-cfg.dot) (Java\_org\_eclipse\_swt\_internal\_gtk\_OS\_g\_1signal\_1handler\_1disconnect)  …  ->12358(c\_os\_12-cfg.dot) (Java\_org\_eclipse\_swt\_internal\_gtk\_OS\_g\_1signal\_1handler\_1disconnect)  ->608(java\_DropTarget\_22-cfg.dot) (onDispose)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data type error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: function definition |

| **Case 197**  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/mrjesen/ImBlockerFabric/issues/28> where it says    Thus, the symptom is mechanism does not work |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c>    Bug is at line 48 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  48(c\_immanager\_11-cfg.dot) (myXFilterEvent)  …  ->44(c\_immanager\_11-cfg.dot) (myXFilterEvent)  ->163(c\_immanager\_11-cfg.dot) (hook)  …  ->80(c\_immanager\_11-cfg.dot) (hook)  ->178(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  …  ->173(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  ->57(java\_LinuxlmManager\_22-cfg.dot) (makeOff)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add if and function call to handle keyboard event  Location: function call + if/branching statement |

| **Case 198**  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/mrjesen/ImBlockerFabric/issues/28> where it says    Thus, the symptom is mechanism does not work |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c>      Bug is at line 48 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  53(c\_immanager\_11-cfg.dot) (myXFilterEvent)  …  ->44(c\_immanager\_11-cfg.dot) (myXFilterEvent)  ->163(c\_immanager\_11-cfg.dot) (hook)  …  ->80(c\_immanager\_11-cfg.dot) (hook)  ->178(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  …  ->173(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  ->57(java\_LinuxlmManager\_22-cfg.dot) (makeOff)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + update variable  Location: function call |

| **Case 199**  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/mrjesen/ImBlockerFabric/issues/29> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/mrjesen/ImBlockerFabric/commit/b2a9c128b66d7d88b4746aeb94b5abbf48caca9c>    Bug is at line 131 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  131(c\_immanager\_11-cfg.dot) (hook)  …  ->56(c\_immanager\_11-cfg.dot) (hook)  ->178(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  …  ->173(c\_immanager\_11-cfg.dot) (Java\_com\_ddwhm\_jesen\_imblocker\_immanager\_linux\_LinuxImManager\_disableIme)  ->57(java\_LinuxlmManager\_22-cfg.dot) (makeOff)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Logic error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add break in the loop  Location: loop + if/branching statement |

| **Case 200**  <https://github.com/MDSplus/mdsplus/commit/707f1656dc68c3c863f4d46d4b5a8ce663f0cdad> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MDSplus/mdsplus/issues/2375> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/MDSplus/mdsplus/commit/707f1656dc68c3c863f4d46d4b5a8ce663f0cdad>    Bug is at line 81 |
| **Bug manifestation:**  From the bug location, backward search until origin in the other language:  988(c\_TreeGetSetShotId\_5-cfg.dot) (GetFileName)  ->1010(c\_TreeSegments\_6-cfg.dot) (begin\_local\_nci)  …  ->1001(c\_TreeSegments\_6-cfg.dot) (begin\_local\_nci)  ->1429(c\_TreeSegments\_6-cfg.dot) (\_TreeXNciMakeSegment)  …  ->1418(c\_TreeSegments\_6-cfg.dot) (\_TreeXNciMakeSegment)  ->2927(c\_TreeSegments\_6-cfg.dot) (\_TreeMakeSegment)  …  ->2923(c\_TreeSegments\_6-cfg.dot) (\_TreeMakeSegment)  ->2934(c\_TreeSegments\_6-cfg.dot) (TreeMakeSegment)  …  ->2930(c\_TreeSegments\_6-cfg.dot) (TreeMakeSegment)  ->2833(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_makeSegment)  …  ->2819(c\_mdsobjects\_2-cfg.dot) (Java\_MDSplus\_TreeNode\_makeSegment)  ->51(java\_TREE\_Test\_19-cfg.dot) (construct)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | |  |  |  | |  |  |  |   Data/control flow direction:   * Java->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Special case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: check the high 16 bit is 0  Location: if/branching statement + assignment |