| **Case 1** <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/scipy/scipy/pull/16649 , which points to an earlier report at <https://github.com/scipy/scipy/pull/12473> where it says    Thus, the symptom is segfault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a>    Bug is at line 194 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  171(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 166(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 1052(py\_\_fitpack\_impl\_14-cfg.dot) (bisplev)  -> 1050(py\_\_fitpack\_impl\_14-cfg.dot) (bisplev)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | … | NAX |  |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + type conversion  Location: if/branching statement + type-conversion expression assignment |

| **Case 2**  <https://github.com/scipy/scipy/commit/60dc9730d5652f0632cd43caef437f01a734e374> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/13095>, which points to an earlier report at <https://github.com/scipy/scipy/issues/3203> where it says    Thus, the symptom is incorrect results/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/60dc9730d5652f0632cd43caef437f01a734e374>    Bug is at line 763 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  Path:  820(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  -> 818(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  -> 599(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 594(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 593(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 591(py\_interpolation\_6-cfg.dot) (affine\_transform)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Numerical and algorithmic errors |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Introducing a new parameter.  Location: if/branching statement |

| **Case 3** <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/19289> where it says    Thus, the symptom is memory leak. |
| **Bug location:**  In the original code, after obtaining dtype, Py\_INCREF was used to increase the reference count of dtype->singleton, but there was no corresponding decrease in the reference count of dtype itself. This could lead to a memory leak, as the dtype object would not be properly released after the function's completion.  In the revised code, after increasing the reference count of dtype->singleton, the reference count of dtype is immediately decreased using Py\_DECREF(dtype). This ensures that the dtype object is correctly released when no longer needed, thus avoiding memory leaks.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/2b18f72ad0784abdbc2eb302eb84511f62042a9b>    Bug is at line 4056 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  4056(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 4055(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 4052(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 4051(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 4352(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  …  -> 4319(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  -> 1590(c\_descriptor\_24-cfg.dot) (PyArray\_DescrConverter)  -> 1589(c\_descriptor\_24-cfg.dot) (PyArray\_DescrConverter)  -> 1539(c\_descriptor\_23-cfg.dot) (\_convert\_from\_any)  …  -> 1478(c\_descriptor\_23-cfg.dot) (\_convert\_from\_any)  -> 1589(c\_descriptor\_24-cfg.dot) (PyArray\_DescrConverter)  -> 1586(c\_descriptor\_24-cfg.dot) (PyArray\_DescrConverter)  -> 4319(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  -> 4318(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  -> 4313(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  -> 4311(c\_ufunc\_object\_57-cfg.dot) (\_get\_dtype)  -> 4051(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  …  -> 4028(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 237(c\_override\_5-cfg.dot) (PyUFunc\_CheckOverride)  ..  -> 229(c\_override\_5-cfg.dot) (PyUFunc\_CheckOverride)  -> 81(c\_override\_1-cfg.dot) (get\_array\_ufunc\_overrides)  …  -> 25(c\_override\_1-cfg.dot) (get\_array\_ufunc\_overrides)  -> 229(c\_override\_5-cfg.dot) (PyUFunc\_CheckOverride)  …  -> 209(c\_override\_5-cfg.dot) (PyUFunc\_CheckOverride)  -> 4028(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  …  -> 3895(c\_ufunc\_object\_55-cfg.dot) (PyUFunc\_GenericReduction)  -> 5739(c\_ufunc\_object\_80-cfg.dot) (ufunc\_accumulate)  -> 5735(c\_ufunc\_object\_80-cfg.dot) (ufunc\_accumulate)  -> 1309(py\_test\_ufunc\_113-cfg.dot) (test\_object\_array\_accumulate\_inplace)  -> 1307(py\_test\_ufunc\_113-cfg.dot) (test\_object\_array\_accumulate\_inplace)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_XDECREF | Foreign | Parsing arguments and building values | | PyArray\_FromAny | NAX | Data conversion | | PyArray\_DescrFromType | NAX | Data conversion |   Data/control flow direction:   * Python->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: Changing the foreign function  Location: near assignment |

| **Case 4** <https://github.com/numpy/numpy/commit/028bdf88e6c71a705d0c6bfd55d1117c7f44f6aa> |
| --- |
| **Symptoms:**  No symptom description/Not available |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/028bdf88e6c71a705d0c6bfd55d1117c7f44f6aa>    Bug is at line 558 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  555(c\_ctors\_14-cfg.dot) (PyArray\_AssignFromCache\_Recursive)  -> 554(c\_ctors\_14-cfg.dot) (PyArray\_AssignFromCache\_Recursive)  …  -> 462(c\_ctors\_14-cfg.dot) (PyArray\_AssignFromCache\_Recursive)  -> 598(c\_ctors\_15-cfg.dot) (PyArray\_AssignFromCache)  …  -> 585(c\_ctors\_15-cfg.dot) (PyArray\_AssignFromCache)  -> 1762(c\_ctors\_28-cfg.dot) (PyArray\_FromAny)  …  -> 1531(c\_ctors\_28-cfg.dot) (PyArray\_FromAny)  -> 2122(c\_mapping\_21-cfg.dot) (\_nonzero\_indices)  …  -> 2108(c\_mapping\_21-cfg.dot) (\_nonzero\_indices)  -> 617(c\_mapping\_8-cfg.dot) (prepare\_index)  …  -> 379(c\_mapping\_8-cfg.dot) (prepare\_index)  -> 3265(c\_mapping\_27-cfg.dot) (PyArray\_MapIterArrayCopyIfOverlap)  …  -> 3255(c\_mapping\_27-cfg.dot) (PyArray\_MapIterArrayCopyIfOverlap)  -> 5725(c\_ufunc\_object\_81-cfg.dot) (ufunc\_at)  …  -> 5644(c\_ufunc\_object\_81-cfg.dot) (ufunc\_at)  -> 1683(py\_test\_ufunc\_153-cfg.dot) (test\_inplace\_fancy\_indexing)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PySequence\_Length | Foreign | Sequence Protocol | | PyErr\_SetString | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_Pack | NAX | Array API | | PyArray\_BYTES | NAX | Array API | | PyArray\_STRIDES | NAX | Array API | | PyArray\_SETITEM | NAX | Array API |   Data/control flow direction:   * Python->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 5** <https://github.com/numpy/numpy/commit/565fe77f9ed8ed5720b42ddc3dc30947a1f3e9ae> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/17430> 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/numpy/numpy/commit/565fe77f9ed8ed5720b42ddc3dc30947a1f3e9ae>    Bug is at line 391,397,401,409 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  : 391(c\_arrayfunction\_override\_10-cfg.dot) (array\_implement\_c\_array\_function\_creation)  -> 390(c\_arrayfunction\_override\_10-cfg.dot) (array\_implement\_c\_array\_function\_creation)  …  -> 366(c\_arrayfunction\_override\_10-cfg.dot) (array\_implement\_c\_array\_function\_creation)  -> 2943(c\_multiarraymodule\_57-cfg.dot) (array\_arange)  ..  -> 2925(c\_multiarraymodule\_57-cfg.dot) (array\_arange)  -> 2788(py\_test\_multiarray\_423-cfg.dot) (test\_partition\_unicode\_kind)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyDict\_DelItem | Foreign | Dictionary Objects | | PyDict\_Contains | Foreign | Dictionary Objects | | PyTuple\_Pack | Foreign | Tuple Objects | | PyImport\_Import | Foreign | Importing Modules | | PyObject\_GetAttrString | Foreign | Object Protocol) |   Data/control flow direction:   * Python->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 some foreign functions  Location: before return statements |

| **Case 6** <https://github.com/numpy/numpy/commit/52c9e6f2e3e72081576fc7f629fe3eb0c67b143b> |
| --- |
| **Symptoms:**  Memory leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/52c9e6f2e3e72081576fc7f629fe3eb0c67b143b>    Bug is at line 211 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  211(c\_reduction\_2-cfg.dot) (conform\_reduce\_result)  ...  -> 86(c\_reduction\_2-cfg.dot) (conform\_reduce\_result)  -> 262(c\_reduction\_3-cfg.dot) (PyArray\_CreateReduceResult)  ...  -> 241(c\_reduction\_3-cfg.dot) (PyArray\_CreateReduceResult)  ->498(c\_reduction\_6-cfg.dot) (PyUFunc\_ReduceWrapper)  ...  -> 450(c\_reduction\_6-cfg.dot) (PyUFunc\_ReduceWrapper)  -> 3588(c\_ufunc\_object\_57-cfg.dot) (PyUFunc\_Reduce)  ...  -> 3577(c\_ufunc\_object\_57-cfg.dot) (PyUFunc\_Reduce)  -> 4609(c\_ufunc\_object\_60-cfg.dot) (PyUFunc\_GenericReduction)  ...  -> 4394(c\_ufunc\_object\_60-cfg.dot) (PyUFunc\_GenericReduction)  -> 5545(c\_ufunc\_object\_80-cfg.dot) (ufunc\_reduceat)  ...  -> 5532(c\_ufunc\_object\_80-cfg.dot) (ufunc\_reduceat)  -> 3118(py\_test\_umath\_777-cfg.dot) (test\_reduceat\_empty)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PyErr\_Format | Foreign | Exception Handling | | PyArray\_DATA | NAX | Array API | | PyArray\_SetWritebackIfCopyBase | NAX | Array API | | PyArray\_NewLikeArray | NAX | Array API | | PyArray\_SetBaseObject | NAX | Array API | | PyArray\_FLAGS | NAX | Array API | | PyArray\_DESCR | NAX | Array API | | PyArray\_NDIM | NAX | Array API |   Data/control flow direction:   * Python->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: Delete the foreign function  Location: if/branching statement |

| **Case 7** <https://github.com/numpy/numpy/commit/a9652077be95f83f56c9b77e7ad1ed7710516626> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/16351> 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/numpy/numpy/commit/a9652077be95f83f56c9b77e7ad1ed7710516626>  Modifications in the raise\_memory\_error Function  In the original version of the function, an error would lead to a jump to the fail label, followed by the execution of PyErr\_WriteUnraisable(NULL);, and then returning PyErr\_NoMemory();. If an error occurred before the creation of exc\_value (for example, if shape was created successfully but exc\_value failed), the allocated resources (like shape) might not have been properly released.  In the modified version, the function's return type has been changed to void, and at the fail label, it directly calls PyErr\_NoMemory(); without returning a value. This means the function no longer returns a new reference, thereby avoiding additional reference count issues in the exception handling path. These changes ensure that no unhandled resources are left in the event of an exception, thus fixing the potential memory leak.  Modifications in the PyArray\_NewFromDescr\_int Function  The original code directly called return raise\_memory\_error(fa->nd, fa->dimensions, descr); when a memory allocation failed, but did not properly clean up fa before jumping to raise\_memory\_error. This could lead to the leakage of resources allocated to fa.  In the revised version, when memory allocation fails, the code calls the raise\_memory\_error function and then jumps to the fail label. At the fail label, the code executes Py\_DECREF(fa);, ensuring the release of resources occupied by the fa object. These changes fix the potential memory leak in the original code by ensuring that all allocated resources are correctly released in all error scenarios.    Bug is at line 902 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  897(c\_ctors\_21-cfg.dot) (raise\_memory\_error)  ...  -> 872(c\_ctors\_21-cfg.dot) (raise\_memory\_error)  -> 1082(c\_ctors\_22-cfg.dot) (PyArray\_NewFromDescr\_int)  ...  -> 912(c\_ctors\_22-cfg.dot) (PyArray\_NewFromDescr\_int)  -> 399(c\_methods\_15-cfg.dot) (PyArray\_GetField)  ...  -> 353(c\_methods\_15-cfg.dot) (PyArray\_GetField)  -> 424(c\_methods\_16-cfg.dot) (array\_getfield)  ...  -> 409(c\_methods\_16-cfg.dot) (array\_getfield)  -> 576(py\_records\_42-cfg.dot) (field)  ...  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_WriteUnraisable | Foreign | Exception Handling | | PyErr\_SetObject | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_IntTupleFromIntp | NAX | Array API |   Data/control flow direction:   * Python->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: Changing error raising function  Location: return statement |

| **Case 8** <https://github.com/numpy/numpy/commit/ce3d79a7c41e6be0a3ad0e8e3214f96fe29c77f9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/16351> 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/numpy/numpy/commit/a9652077be95f83f56c9b77e7ad1ed7710516626>    Bug is at line 1872 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1873(c\_multiarraymodule\_38-cfg.dot) (array\_empty\_like)  ...  -> 1851(c\_multiarraymodule\_38-cfg.dot) (array\_empty\_like)  -> 382(py\_numeric\_11-cfg.dot) (full\_like)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyArray\_NewLikeArrayWithShape | NAX | Array API |   Data/control flow direction:   * Python->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: Free the variable  Location: After the variable which need to be free after use |

| **Case 9** <https://github.com/numpy/numpy/commit/fc2518ba6b11fc52b0ff477b9e83576be90562d8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/16336> 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/numpy/numpy/commit/a9652077be95f83f56c9b77e7ad1ed7710516626>  First Modification - Addition of Py\_INCREF(str); in \_ADDDOC Macro:  This modification is made within the \_ADDDOC macro, which is used to add a docstring to different types of Python objects. The Py\_INCREF(str); line is added to increment the reference count of the str object (which is the new docstring). This is important because doc is being assigned the value of docstr, which is a pointer to the internal buffer of str. The Py\_INCREF ensures that the str object is not prematurely deallocated while its buffer is in use, preventing potential use-after-free issues.  Second Modification - Addition of Py\_DECREF(doc\_attr); in the Else Block:  This change occurs in the else block handling objects that do not fall into the previous categories (like function, type, member, etc.). The Py\_DECREF(doc\_attr); is added after the comparison between the current docstring (doc\_attr) and the new one (str). This is necessary because PyObject\_GetAttrString increments the reference count of the returned object (doc\_attr). If the current and new docstrings are different, doc\_attr needs to be decremented to prevent a memory leak, as the reference is no longer needed.  Third Modification - Removal of Py\_INCREF(str); After \_ADDDOC Macro Calls:  The removal of this Py\_INCREF(str); line at the end of the function is related to the first modification. Initially, str's reference count was incremented here, after all \_ADDDOC calls. However, with the first modification, the Py\_INCREF(str); is now done within each \_ADDDOC call when needed. Therefore, this separate increment at the end of the function is redundant and could lead to a reference count imbalance, potentially preventing str from being deallocated properly when it's no longer needed.    Bug is at line ? |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1477(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1476(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1469(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1465(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1461(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1457(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1453(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1440(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1439(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1435(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1431(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1428(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 1418(c\_compiled\_base\_17-cfg.dot) (arr\_add\_docstring)  -> 3375(py\_test\_function\_base\_722-cfg.dot) (test\_add\_same\_docstring)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_INCREF | Foreign | Reference Counting | | PyObject\_SetAttrString | Foreign | Object Protocol | | PyErr\_SetString | Foreign | Exception Handling | | PyObject\_GetAttrString | Foreign | Object Protocol | | Py\_TYPE | Foreign | Common Object Structures | | PyErr\_Occurred | Foreign | Exception Handling |   Data/control flow direction:   * Python->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: Increase/decrease the reference counting before/after used  Location: if/branching statement |

| **Case 10**  <https://github.com/numpy/numpy/commit/8cb86fd7b454b40c7b822146d5e26c55fdc183ec> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/16276> where it says    Thus, the symptom is error message (type error). |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/8cb86fd7b454b40c7b822146d5e26c55fdc183ec>    Bug is at line 120 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  153(c\_methods\_4-cfg.dot) (array\_take)  ...  -> 149(c\_methods\_4-cfg.dot) (array\_take)  -> 306(c\_item\_selection\_1-cfg.dot) (PyArray\_TakeFrom)  ...  -> 190(c\_item\_selection\_1-cfg.dot) (PyArray\_TakeFrom)  -> 149(c\_methods\_4-cfg.dot) (array\_take)  ...  -> 133(c\_methods\_4-cfg.dot) (array\_take)  -> 994(py\_test\_indexing\_203-cfg.dot) (\_get\_multi\_index)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyArray\_Return | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Incorrect output |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: return statement + if/branching statement |

| **Case 1**1 <https://github.com/numpy/numpy/commit/21e796e159dd4865a265b94a044ddb144e4e0af1> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/scipy/scipy/pull/16649 , which points to an earlier report at <https://github.com/numpy/numpy/pull/15164> where it says    Thus, the Type error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  https://github.com/numpy/numpy/commit/21e796e159dd4865a265b94a044ddb144e4e0af1    Bug is at line 1472 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1472(c\_methods\_44-cfg.dot) (array\_searchsorted)  -> 1469(c\_methods\_44-cfg.dot) (array\_searchsorted)  -> 1068(py\_histograms\_21-cfg.dot) (histogramdd)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyArray\_Return | NAX | Array API | | PyArray\_SearchSorted | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Incorrect argument |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument  Location: variable initialization |

| **Case 12** <https://github.com/numpy/numpy/commit/6ed9365aadf992087b26ec30214dfec151ac22a1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14769> , which points to an earlier report at <https://github.com/numpy/numpy/issues/14767> where it says    Thus, the value error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/6ed9365aadf992087b26ec30214dfec151ac22a1>    Bug is at line 1628 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1628(c\_multiarraymodule\_35-cfg.dot) (\_array\_fromobject)  -> 1627(c\_multiarraymodule\_35-cfg.dot) (\_array\_fromobject)  …  -> 724(py\_test\_regression\_101-cfg.dot) (test\_noncommutative\_reduce\_accumulate)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyLong\_AsLong | Foreign | Integer Objects | | PyDict\_GetItem | Foreign | Dictionary Objects | | PyTuple\_GET\_ITEM | Foreign | Tuple Objects | | PyTuple\_GET\_SIZE | Foreign | Tuple Objects | | PyErr\_SetString | Foreign | Exception Handling | | PyArray\_IS\_C\_CONTIGUOUS | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_NewCopy | NAX | Array API | | PyArray\_CheckExact | NAX | Array API |   Data/control flow direction:   * Python->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: Data type change  Location:NAX function |

| **Case 13** <https://github.com/numpy/numpy/commit/3d31770c61ea2412267c233d38ccc33d5d3a0610> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14585> where it says    Thus, the symptom is crash or invalid results |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  https://github.com/numpy/numpy/commit/3d31770c61ea2412267c233d38ccc33d5d3a0610    Bug is at line 963 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  977(c\_compiled\_base\_13-cfg.dot) (ravel\_multi\_index\_loop)  …  -> 934(c\_compiled\_base\_13-cfg.dot) (ravel\_multi\_index\_loop)  -> 1118(c\_compiled\_base\_14-cfg.dot) (arr\_ravel\_multi\_index)  …  -> 1004(c\_compiled\_base\_14-cfg.dot) (arr\_ravel\_multi\_index)  -> 106(py\_test\_index\_tricks\_4-cfg.dot) (test\_big\_indices)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Division by Zero |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + loop statement |

| **Case 14** <https://github.com/numpy/numpy/commit/f786041db9697f58b087e18198561db8b28235c4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14240> where it says    Thus, the symptom is incorrect output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/f786041db9697f58b087e18198561db8b28235c4>    Bug is at line 3847 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  3847(c\_ctors\_55-cfg.dot) (PyArray\_FromBuffer)  ...  -> 3775(c\_ctors\_55-cfg.dot) (PyArray\_FromBuffer)  -> 2156(c\_multiarraymodule\_45-cfg.dot) (array\_frombuffer)  ...  -> 2139(c\_multiarraymodule\_45-cfg.dot) (array\_frombuffer)  -> 49(py\_testFlat\_4-cfg.dot) (testProcess3D)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyIter\_Next | Foreign | Iterator Protocol | | PyErr\_SetString | Foreign | Exception Handling | | PyObject\_LengthHint | Foreign | Object Protocol | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_NewFromDescr | NAX | Array API | | PyDataType\_REFCHK | NAX | Python types and C-structures | | PyDataType\_ISUNSIZED | NAX | Python types and C-structures |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Division by Zero |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + Foreign function |

| **Case 15** <https://github.com/numpy/numpy/commit/4246ce2a391314acc1da90d56ff4f995d45e18a9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14585>, which points to an earlier report at <https://github.com/numpy/numpy/issues/14228> where it says    Thus, the symptom is memory error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/4246ce2a391314acc1da90d56ff4f995d45e18a9>    Bug is at line 1459 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1459(c\_citem\_selection\_55-cfg.dot) (PyArray\_LexSort)  ...  -> 1388(c\_citem\_selection\_55-cfg.dot) (PyArray\_LexSort)  -> 3464(c\_multiarraymodule\_45-cfg.dot) (array\_lexsort)  ...  -> 3450(c\_multiarraymodule\_45-cfg.dot) (array\_lexsort)  -> 49(py\_test\_regression\_4-cfg.dot) (test\_lexsort\_zerolen\_custom\_strides)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_Format | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_SetString | Foreign | Exception Handling | | PySequence\_GetItem | Foreign | Sequence Protocol | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_NDIM | NAX | Array API | | PyArray\_DIMS | NAX | Array API | | PyArray\_FROM\_O | NAX | Array API | | PyDataType\_FLAGCHK | NAX | Python types and C-structures | | PyArray\_DESCR | NAX | Array API | | PyArray\_CompareLists | NAX | Array API | | PyArray\_malloc | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  The root cause appears to be that the code does not properly handle the situation of attempting to create a new array from an empty array (where the total number of elements is 0). This may lead to the use of unreliable dimensional information for memory allocation, resulting in a memory error.  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 16** <https://github.com/numpy/numpy/commit/4246ce2a391314acc1da90d56ff4f995d45e18a9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14585>, which points to an earlier report at <https://github.com/numpy/numpy/issues/14228> where it says    Thus, the symptom is memory error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/4246ce2a391314acc1da90d56ff4f995d45e18a9>    Bug is at line 1459 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1459(c\_citem\_selection\_55-cfg.dot) (PyArray\_LexSort)  ...  -> 1388(c\_citem\_selection\_55-cfg.dot) (PyArray\_LexSort)  -> 3464(c\_multiarraymodule\_45-cfg.dot) (array\_lexsort)  ...  -> 3450(c\_multiarraymodule\_45-cfg.dot) (array\_lexsort)  -> 49(py\_test\_regression\_4-cfg.dot) (test\_lexsort\_zerolen\_custom\_strides)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_Format | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_SetString | Foreign | Exception Handling | | PySequence\_GetItem | Foreign | Sequence Protocol | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_NDIM | NAX | Array API | | PyArray\_DIMS | NAX | Array API | | PyArray\_FROM\_O | NAX | Array API | | PyDataType\_FLAGCHK | NAX | Python types and C-structures | | PyArray\_DESCR | NAX | Array API | | PyArray\_CompareLists | NAX | Array API | | PyArray\_malloc | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  The root cause appears to be that the code does not properly handle the situation of attempting to create a new array from an empty array (where the total number of elements is 0). This may lead to the use of unreliable dimensional information for memory allocation, resulting in a memory error.  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 17** <https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/14143> , which points to an earlier report at <https://github.com/numpy/numpy/issues/14077> where it says    Thus, the symptom is incorrect output. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24>    Bug is at line 2275 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2275(c\_[datetime](https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24#diff-c32b1134e6b616b5aca0e41218075be0ae2cb509a03e46b1716e1988a93fafd8)\_54-cfg.dot) (convert\_pydatetime\_to\_datetimestruct)  ...  -> 2103(c\_[datetime](https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24#diff-c32b1134e6b616b5aca0e41218075be0ae2cb509a03e46b1716e1988a93fafd8)\_21-cfg.dot) (convert\_pydatetime\_to\_datetimestruct)  -> 2338(c\_[datetime](https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24#diff-c32b1134e6b616b5aca0e41218075be0ae2cb509a03e46b1716e1988a93fafd8)\_21-cfg.dot) (get\_tzoffset\_from\_pytzinfo)  ...  -> 2317(c\_[datetime](https://github.com/numpy/numpy/commit/608329acac78b98e9c1044dca23a4f287d639b24#diff-c32b1134e6b616b5aca0e41218075be0ae2cb509a03e46b1716e1988a93fafd8)\_21-cfg.dot) (get\_tzoffset\_from\_pytzinfo)  -> 1562(c\_datetime\_strings\_58-cfg.dot) (array\_datetime\_as\_string)  …  -> 1338(c\_datetime\_strings\_58-cfg.dot) (array\_datetime\_as\_string)  -> 792(py\_test\_datetime\_13-cfg.dot) (test\_datetime\_array\_str)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyObject\_CallMethod | Foreign | Call Protocol | | PyErr\_Clear | Foreign | Exception Handling | | PyObject\_GetAttrString | Foreign | Object Protocol | | PyObject\_HasAttrString | Foreign | Object Protocol | | PyFloat\_AsDouble | Foreign | Floating Point Objects | | Py\_DECREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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: Use correct foreign function  Location: assignment statement + foreign function |

| **Case 18** <https://github.com/numpy/numpy/commit/842079e235dfac64cf99059286e295b307d073fc> |
| --- |
| **Symptoms:**  No description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/842079e235dfac64cf99059286e295b307d073fc>    Bug is at line 139 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  139(c\_[shape](https://github.com/numpy/numpy/commit/842079e235dfac64cf99059286e295b307d073fc#diff-4e0e06f243ddff9085e315e87fc4ab93becf2d6272b5dc0fdc03ae5fad14f5dd)\_51-cfg.dot) (PyArray\_Resize)  ...  -> 41(c\_[shape](https://github.com/numpy/numpy/commit/842079e235dfac64cf99059286e295b307d073fc#diff-4e0e06f243ddff9085e315e87fc4ab93becf2d6272b5dc0fdc03ae5fad14f5dd)\_51-cfg.dot) (PyArray\_Resize)  -> 1176(c\_methods\_21-cfg.dot) (array\_resize)  ...  -> 1146(c\_methods\_21-cfg.dot) (array\_resize)  -> 124(py\_test\_datetime\_13-cfg.dot) (\_\_new\_\_)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_SetString | Foreign | Exception Handling | | PyArray\_BYTES | NAX | Array API | | PyDataType\_FLAGCHK | NAX | Python types and C-structures | | PyArray\_ISWRITEABLE | NAX | Array API | | PyArray\_DATA | NAX | Array API | | PyArray\_SIZE | NAX | Array API |   Data/control flow direction:   * Python->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 19** <https://github.com/numpy/numpy/commit/52d173d9087f6fb09994b433a127f58d19875c98> |
| --- |
| **Symptoms:**  No description. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/52d173d9087f6fb09994b433a127f58d19875c98>    Bug is at line 2108 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2108(c\_methods\_56-cfg.dot) (PyArray\_Dump)  …  -> 2167(c\_methods\_56-cfg.dot) (PyArray\_Dump)  -> 2167(c\_methods\_56-cfg.dot) (array\_dump)  …  -> 2158(c\_methods\_56-cfg.dot) (array\_dump)  -> 241(py\_\_methods\_16-cfg.dot) (\_dump)  -> 240(py\_\_methods\_16-cfg.dot) (\_dump)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyImport\_ImportModule | Foreign | Importing Modules | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  File I/O error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Move the function to python  Location: if/branching statement + foreign function |

| **Case 20** <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/scipy/scipy/pull/16649 , which points to an earlier report at <https://github.com/scipy/scipy/pull/12473> where it says    Thus, the symptom is segfault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a>    Bug is at line 194 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  171(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 166(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 1052(py\_\_fitpack\_impl\_14-cfg.dot) (bisplev)  -> 1050(py\_\_fitpack\_impl\_14-cfg.dot) (bisplev)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | … | NAX |  |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + type conversion  Location: if/branching statement + type-conversion expression assignment |

| **Case 21** <https://github.com/numpy/numpy/commit/1b144ad9d45647960c009ff4fe4e1b18958ff997> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/12696> where it says    Thus, the symptom is memory leak. |
| **Bug location:**  The newly added line Py\_DECREF(descr); seems to decrease the reference count of descr under an additional condition, possibly to prevent memory leakage. After creating a new descr object using PyObject\_CallFunctionObjArgs, the original descr object might no longer be needed, so its reference count is decreased to avoid memory leaks.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/1b144ad9d45647960c009ff4fe4e1b18958ff997>    Bug is at line 1413 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1413(c\_ctors\_24-cfg.dot) (\_array\_from\_buffer\_3118)  -> 1412(c\_ctors\_24-cfg.dot) (\_array\_from\_buffer\_3118)  -> 1398(c\_ctors\_24-cfg.dot) (\_array\_from\_buffer\_3118)  -> 1386(c\_ctors\_24-cfg.dot) (\_array\_from\_buffer\_3118)  -> 896(py\_\_internal\_96-cfg.dot) (npy\_ctypes\_check)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_CallFunctionObjArgs | Foreign | Call Protocol | | PyErr\_Warn | Foreign | Exception Handling | | PyErr\_Format | Foreign | Exception Handling | | MemoryView objects | Foreign | MemoryView objects |   Data/control flow direction:   * C-> Python |
| **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 foreign function  Location: assignment statement + foreign function |

| **Case 22** <https://github.com/numpy/numpy/commit/2b05f3e38431842ff06df9b2958d22c5a0588767> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/12696> where it says    Thus, the symptom is reference count error.. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/2b05f3e38431842ff06df9b2958d22c5a0588767>    Bug is at line 2499 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2499(c\_ctors\_31-cfg.dot) (PyArray\_FromInterface)  …  -> 2289(c\_ctors\_31-cfg.dot) (PyArray\_FromInterface)  -> 1619(c\_ctors\_25-cfg.dot) (PyArray\_GetArrayParamsFromObject)  …  -> 1539(c\_ctors\_25-cfg.dot) (PyArray\_GetArrayParamsFromObject)  -> 263(c\_convert\_datatype\_3-cfg.dot) (PyArray\_AdaptFlexibleDType)  …  -> 146(c\_convert\_datatype\_3-cfg.dot) (PyArray\_AdaptFlexibleDType)  -> 1169(c\_convert\_datatype\_18-cfg.dot) (PyArray\_PromoteTypes)  …  -> 1002(c\_convert\_datatype\_18-cfg.dot) (PyArray\_PromoteTypes)  -> 1327(c\_convert\_datatype\_19-cfg.dot) (PyArray\_PromoteTypeSequence)  …  -> 1315(c\_convert\_datatype\_19-cfg.dot) (PyArray\_PromoteTypeSequence)  -> 1790(c\_convert\_datatype\_24-cfg.dot) (PyArray\_ResultType)  …  -> 1705(c\_convert\_datatype\_24-cfg.dot) (PyArray\_ResultType)  -> 2456(c\_nditer\_constr\_42-cfg.dot) (npyiter\_get\_common\_dtype)  …  -> 2410(c\_nditer\_constr\_42-cfg.dot) (npyiter\_get\_common\_dtype)  -> 351(c\_nditer\_constr\_20-cfg.dot) (NpyIter\_AdvancedNew)  …  -> 103(c\_nditer\_constr\_20-cfg.dot) (NpyIter\_AdvancedNew)  -> 1809(c\_ufunc\_object\_39-cfg.dot) (execute\_fancy\_ufunc\_loop)  …  -> 1762(c\_ufunc\_object\_39-cfg.dot) (execute\_fancy\_ufunc\_loop)  -> 3239(c\_ufunc\_object\_52-cfg.dot) (PyUFunc\_GenericFunction)  …  -> 3099(c\_ufunc\_object\_52-cfg.dot) (PyUFunc\_GenericFunction)  -> 4686(c\_ufunc\_object\_60-cfg.dot) (ufunc\_generic\_call)  …  -> 4667(c\_ufunc\_object\_60-cfg.dot) (ufunc\_generic\_call)  -> 5401(c\_ufunc\_object\_75-cfg.dot) (ufunc\_outer)  …  -> 5312(c\_ufunc\_object\_75-cfg.dot) (ufunc\_outer)  -> 1378(py\_test\_core\_143-cfg.dot) (test\_TakeTransposeInnerOuter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyObject\_AsReadBuffer | Foreign | Old Buffer Protocol | | PyLong\_AsLongLong | Foreign | Integer Objects | | PyErr\_SetString | Foreign | Exception Handling | | PyDict\_GetItemString | Foreign | Dictionary Objects | | PyObject\_AsWriteBuffer | Foreign | Old Buffer Protocol | | PyBuffer\_Release | Foreign | Buffer Protocol | | PyObject\_GetBuffer | Foreign | Buffer Protocol | | PyTuple\_GET\_ITEM | Foreign | Tuple Objects | | PyObject\_IsTrue | Foreign | Object Protocol | | PyLong\_AsVoidPtr | Foreign | nteger Objects | | PyArray\_NewFromDescrAndBase | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Invalid pointer |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Reset the pointer  Location: assignment statement |

| **Case 23** <https://github.com/numpy/numpy/commit/74f3d07ab0bcfd63f42f62b26eeb6ce68efd4f21> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/12814>, which points to an earlier report at <https://github.com/numpy/numpy/issues/12806> where it says    Thus, the symptom is runtime warning. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/74f3d07ab0bcfd63f42f62b26eeb6ce68efd4f21>    Bug is at line 332 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  : 330(c\_compiled\_base\_6-cfg.dot) (arr\_insert)  …  -> 259(c\_compiled\_base\_6-cfg.dot) (arr\_insert)  -> 1724(py\_function\_base\_32-cfg.dot) (place)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_XDECREF | Foreign | Reference Counting | | Py\_INCREF | Foreign | Reference Counting | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyArray\_ResolveWritebackIfCopy | NAX | Array API | | PyArray\_SIZE | NAX | Array API | | PyArray\_FromAny | NAX | Array API | | PyArray\_FROM\_OTF | NAX | Array API | | PyArray\_DESCR | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Numerical and algorithmic errors |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add the NAX function  Location: After the foreign function |

| **Case 24** <https://github.com/numpy/numpy/commit/85d64e375f53f6028519d3aaad0b421934dcbe24> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/issues/11969> 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/numpy/numpy/commit/85d64e375f53f6028519d3aaad0b421934dcbe24>    Bug is at line 155 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  155(c\_convert\_datatype\_3-cfg.dot) (PyArray\_AdaptFlexibleDType)  ...  -> 146(c\_convert\_datatype\_3-cfg.dot) (PyArray\_AdaptFlexibleDType)  -> 1169(c\_convert\_datatype\_18-cfg.dot) (PyArray\_PromoteTypes)  ...  -> 1002(c\_convert\_datatype\_18-cfg.dot) (PyArray\_PromoteTypes)  -> 1327(c\_convert\_datatype\_19-cfg.dot) (PyArray\_PromoteTypeSequence)  ...  -> 1315(c\_convert\_datatype\_19-cfg.dot) (PyArray\_PromoteTypeSequence)  -> 1790(c\_convert\_datatype\_24-cfg.dot) (PyArray\_ResultType)  ...  -> 1705(c\_convert\_datatype\_24-cfg.dot) (PyArray\_ResultType)  -> 2456(c\_nditer\_constr\_42-cfg.dot) (npyiter\_get\_common\_dtype)  ...  -> 2410(c\_nditer\_constr\_42-cfg.dot) (npyiter\_get\_common\_dtype)  -> 351(c\_nditer\_constr\_20-cfg.dot) (NpyIter\_AdvancedNew)  ...  -> 103(c\_nditer\_constr\_20-cfg.dot) (NpyIter\_AdvancedNew)  -> 1809(c\_ufunc\_object\_39-cfg.dot) (execute\_fancy\_ufunc\_loop)  ...  -> 1762(c\_ufunc\_object\_39-cfg.dot) (execute\_fancy\_ufunc\_loop)  -> 3241(c\_ufunc\_object\_52-cfg.dot) (PyUFunc\_GenericFunction)  ...  -> 3171(c\_ufunc\_object\_52-cfg.dot) (PyUFunc\_GenericFunction)  -> 4722(c\_ufunc\_object\_60-cfg.dot) (ufunc\_generic\_call)  ...  -> 4714(c\_ufunc\_object\_60-cfg.dot) (ufunc\_generic\_call)  -> 5471(c\_ufunc\_object\_76-cfg.dot) (ufunc\_outer)  ...  -> 5382(c\_ufunc\_object\_76-cfg.dot) (ufunc\_outer)  -> 1378(py\_test\_core\_143-cfg.dot) (test\_TakeTransposeInnerOuter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_Occurred | Foreign | Exception Handling | | PyErr\_SetString | Foreign | Exception Handling | | PyArray\_AdaptFlexibleDType | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Numerical and algorithmic errors |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add return statement  Location: if/branching statement + foreign function |

| **Case 25** <https://github.com/numpy/numpy/commit/dea85807c258ded3f75528cce2a444468de93bc1> |
| --- |
| **Symptoms:**  No description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/dea85807c258ded3f75528cce2a444468de93bc1>    Bug is at line 1434 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1434(c\_item\_selection\_13-cfg.dot) (PyArray\_LexSort)  …  -> 1539(c\_item\_selection\_13-cfg.dot) (PyArray\_LexSort)  -> 1598(py\_test\_multiarray\_25-cfg.dot) (test\_sort)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyErr\_NoMemory | Foreign | Exception Handling | | PySequence\_Check | Foreign | Sequence Protocol | | PySequence\_Size | Foreign | Sequence Protocol | | PyArray\_ISBYTESWAPPED | NAX | Array API | | PyDataMem\_FREE | NAX | Array API | | PyDataMem\_NEW | NAX | Array API | | PyArray\_DESCR | NAX | Array API | | PyArray\_ISBYTESWAPPED | NAX | Array API | | PyArray\_DIMS | NAX | Array API | | PyArray\_IterAllButAxis | NAX | Array API | | PyArray\_DATA | NAX | Array API | | PyArray\_SIZE | NAX | Array API | | PyDataType\_FLAGCHK | NAX | Python types and C-structures | | PyArray\_FROM\_O | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument  Location: assignment statement + NAX function |

| **Case 26** <https://github.com/numpy/numpy/commit/2e697d49a393aba2fcfa9991eaab4f420a58b5d5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/13590> where it says    Thus, the symptom is runtime error. |
| **Bug location:**  When the size argument passed to memcpy is zero, the source and/or destination pointers could be null, which is undefined behavior in C.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/2e697d49a393aba2fcfa9991eaab4f420a58b5d5>    Bug is at line 1057, 1067. |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1057(c\_ctors\_18-cfg.dot) (PyArray\_NewFromDescr\_int)  ...  -> 929(c\_ctors\_18-cfg.dot) (PyArray\_NewFromDescr\_int)  -> 620(c\_convert\_11-cfg.dot) (PyArray\_View)  ...  -> 601(c\_convert\_11-cfg.dot) (PyArray\_View)  -> 553(c\_shape\_11-cfg.dot) (PyArray\_Squeeze)  ...  -> 525(c\_shape\_11-cfg.dot) (PyArray\_Squeeze)  -> 230(c\_methods\_8-cfg.dot) (array\_squeeze)  ...  -> 217(c\_methods\_8-cfg.dot) (array\_squeeze)  -> 1785(py\_test\_ufunc\_141-cfg.dot) (test\_structured\_equal)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_Format | Foreign | Exception Handling | | PyArray\_DESCR\_REPLACE | NAX | Array API | | PyDataType\_ISSTRING | NAX | Array API | | PyDataType\_ISFLEXIBLE | NAX | Array API | | PyDataType\_ISUNSIZED | NAX | Array API |   Data/control flow direction:   * Python->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  Location: if/branching statement |

| **Case 27** <https://github.com/numpy/numpy/commit/842970f1aaa710b31ebd27427035b58b265e55a8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/11684> , which points to an earlier report at <https://github.com/numpy/numpy/issues/7165> where it says    Thus, the symptom is Type error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/842970f1aaa710b31ebd27427035b58b265e55a8>    Bug is at line 866 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  866(c\_compiled\_base\_10-cfg.dot) (sequence\_to\_arrays)  ...  -> 843(c\_compiled\_base\_10-cfg.dot) (sequence\_to\_arrays)  -> 1026(c\_compiled\_base\_12-cfg.dot) (arr\_ravel\_multi\_index)  ...  -> 953(c\_compiled\_base\_12-cfg.dot) (arr\_ravel\_multi\_index)  -> 912(py\_test\_indexing\_205-cfg.dot) (\_get\_multi\_index)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PySequence\_GetItem | Foreign | Sequence Protocol | | PyErr\_Format | Foreign | Exception Handling | | PySequence\_Check | Foreign | Sequence Protocol | | PySequence\_Size | Foreign | Sequence Protocol | | PyArray\_FROM\_O | NAX | Array API |   Data/control flow direction:   * Python->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 NAX function to a new function  Location: assignment statement + NAX function |

| **Case 28** <https://github.com/numpy/numpy/commit/8598315e41ed38faf3fb71cdc7c6f62067a9b2e8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/issues/1737> where it says    Thus, the symptom is Type error. |
| **Bug location:**  In the original code, a single variable value, whose type is determined by the larger of npy\_longlong or npy\_clongdouble, is used to store the value of an array element. This approach generally works well, but it has potential issues, especially when dealing with data types larger than npy\_longlong or npy\_clongdouble.  In the revised code, value is replaced with an array of npy\_longlong type, value[4]. This array can accommodate larger data types, such as 128-bit types. By doing so, the code becomes more robust, capable of handling a wider range of data types, not just limited to npy\_longlong or npy\_clongdouble.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/8598315e41ed38faf3fb71cdc7c6f62067a9b2e8>    Bug is at line 430 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  430(c\_convert\_datatype\_11-cfg.dot) (PyArray\_CanCastArrayTo)  ...  -> 407(c\_convert\_datatype\_11-cfg.dot) (PyArray\_CanCastArrayTo)  -> 996(c\_ufunc\_object\_19-cfg.dot) (ufunc\_loop\_matches)  ...  -> 941(c\_ufunc\_object\_19-cfg.dot) (ufunc\_loop\_matches)  -> 1223(c\_ufunc\_object\_22-cfg.dot) (find\_ufunc\_specified\_userloop)  ...  -> 1163(c\_ufunc\_object\_22-cfg.dot) (find\_ufunc\_specified\_userloop)  -> 1530(c\_ufunc\_object\_24-cfg.dot) (find\_specified\_ufunc\_inner\_loop)  ...  -> 1410(c\_ufunc\_object\_24-cfg.dot) (find\_specified\_ufunc\_inner\_loop)  -> 2566(c\_ufunc\_object\_32-cfg.dot) (PyUFunc\_GenericFunction)  ...  -> 2457(c\_ufunc\_object\_32-cfg.dot) (PyUFunc\_GenericFunction)  -> 4075(c\_ufunc\_object\_40-cfg.dot) (ufunc\_generic\_call)  ...  -> 4057(c\_ufunc\_object\_40-cfg.dot) (ufunc\_generic\_call)  -> 4698(c\_ufunc\_object\_54-cfg.dot) (ufunc\_outer)  ...  -> 4619(c\_ufunc\_object\_54-cfg.dot) (ufunc\_outer)  -> 228(py\_timer\_comparison\_12-cfg.dot) (test\_4)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArray\_BYTES | NAX | Array API | | PyArray\_ISNBO | NAX | Array API | | PyArray\_CanCastTypeTo | NAX | Array API | | PyTypeNum\_ISNUMBER | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_DESCR | NAX | Array API |   Data/control flow direction:   * Python->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 data type  Location: if/branching statement + NAX function |

| **Case 29** <https://github.com/numpy/numpy/commit/914c498abefb4f65e160cdf6b030ff8efe8b975c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/issues/1175> where it says    Thus, the symptom is incorrect results/output |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/914c498abefb4f65e160cdf6b030ff8efe8b975c>    Bug is at line 3531 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  3531(c\_ufunc\_object\_38-cfg.dot) (ufunc\_generic\_call)  ...  -> 3498(c\_ufunc\_object\_38-cfg.dot) (ufunc\_generic\_call)  -> 4137(c\_ufunc\_object\_52-cfg.dot) (ufunc\_outer)  ...  -> 4058(c\_ufunc\_object\_52-cfg.dot) (ufunc\_outer)  -> 225(py\_timer\_comparison\_12-cfg.dot) (test\_4)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArray\_XDECREF\_ERR | NAX | Array API | | PyErr\_SetString | Foreign | Exception Handling | | Py\_INCREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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 branch  Location: if/branching statement + Foreign function |

| **Case 30** <https://github.com/numpy/numpy/commit/6e6c5c4fdb6548ef6eae953ce453a1a89c290fc1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/issues/583> where it says    Thus, the symptom is value error. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/6e6c5c4fdb6548ef6eae953ce453a1a89c290fc1>    Bug is at line 6404 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  6404(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6403(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6401(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6397(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6396(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6392(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6498(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6488(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6486(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6485(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6484(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6480(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 230(py\_test\_numeric\_75-cfg.dot) (test\_values)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyObject\_GetIter | Foreign | Object Protocol | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  miss boundary check |
| **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/numpy/numpy/commit/a0e082a087e9667e3805d3be859958a292e8f336> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/issues/925> where it says    Thus, the symptom is error message. |
| **Bug location:**  **Memory Allocation** (sizes = (intp \*)\_pya\_malloc(n\*sizeof(intp));): This line of code is responsible for allocating memory for the sizes array. The key point here is whether the allocated memory is properly initialized. In C, memory allocation does not guarantee that the memory area is zeroed. If the code accesses these memory positions before they are initialized, it might encounter uninitialized values. In the original code, if (sizes == NULL), the program jumps to the fail label. This means that if sizes is null, the code logic would handle this situation, so this is unlikely to be the cause of uninitialized value usage.  **Array Pointer Initializatio**n (ap = (PyArrayObject \*)PyArray\_ContiguousFromAny((PyObject \*)ip, PyArray\_INTP, 0, 0);): This code attempts to convert the input ip into a contiguous NumPy array. If the conversion fails, ap will be NULL. In the original code, if ap == NULL, the program jumps to the fail label. This means that if ap is null, the code logic would handle this situation, so this is unlikely to be the cause of uninitialized value usage.  **Loop Condition Checks** (for (i=0; i<m; i++) { ... }): This loop includes multiple condition checks and memory accesses. In the original code, the loop variable i is correctly initialized, and multiple places within the loop access or modify arrays and other data structures. If any of these data structures are not properly initialized before use, it might lead to problems.  **Array Operations** (memmove(ret\_data, mps[mi]->data+offset, elsize);): Memory move operations are performed here. It's necessary to ensure that the memory pointed to by mps[mi]->data and ret\_data is initialized, and that offset and elsize are valid. If any of these conditions are not met, it could lead to the use of uninitialized values. This part seems to be the most likely place for uninitialized value usage, especially if the calculation of offset depends on previous operations that involve uninitialized values.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/a0e082a087e9667e3805d3be859958a292e8f336>    Bug is at line 2431 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  6404(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6403(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6401(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6397(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6396(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6392(c\_multiarraymodule\_124-cfg.dot) (PyArray\_FromIter)  -> 6498(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6488(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6486(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6485(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6484(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 6480(c\_multiarraymodule\_125-cfg.dot) (array\_fromiter)  -> 230(py\_test\_numeric\_75-cfg.dot) (test\_values)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | Py\_INCREF | Foreign | Reference Counting | | PyArray\_SIZE | NAX | Array API | | PyArray\_FromArray | NAX | Array API | | PyArray\_NBYTES | NAX | Array API | | PyArray\_CompareLists | NAX | Array API | | PyArray\_ContiguousFromAny | NAX | Array API | | PyArray\_ConvertToCommonType | NAX | Array API |   Data/control flow direction:   * Python->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 NAX function  Location: function call |

| **Case 32** <https://github.com/numpy/numpy/commit/ebf8dce3a385f9da5b1e769cb1c41f8f928dfedf> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/commit/ebf8dce3a385f9da5b1e769cb1c41f8f928dfedf> 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/numpy/numpy/commit/ebf8dce3a385f9da5b1e769cb1c41f8f928dfedf>    Bug is at line 2146, 2152 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2146(c\_multiarraymodule\_47-cfg.dot) (PyArray\_CanCoerceScalar)  ...  -> 2132(c\_multiarraymodule\_47-cfg.dot) (PyArray\_CanCoerceScalar)  -> 1113(c\_ufuncobject\_42-cfg.dot) (select\_types)  ...  -> 1046(c\_ufuncobject\_42-cfg.dot) (select\_types)  -> 1395(c\_ufuncobject\_47-cfg.dot) (construct\_arrays)  ...  -> 1304(c\_ufuncobject\_47-cfg.dot) (construct\_arrays)  -> 1911(c\_ufuncobject\_50-cfg.dot) (construct\_loop)  ...  -> 1834(c\_ufuncobject\_50-cfg.dot) (construct\_loop)  -> 1994(c\_ufuncobject\_51-cfg.dot) (PyUFunc\_GenericFunction)  -> 3365(c\_ufuncobject\_60-cfg.dot) (ufunc\_generic\_call)  -> 3361(c\_ufuncobject\_60-cfg.dot) (ufunc\_generic\_call)  -> 3346(c\_ufuncobject\_60-cfg.dot) (ufunc\_generic\_call)  -> 3973(c\_ufuncobject\_72-cfg.dot) (ufunc\_outer)  ...  -> 3913(c\_ufuncobject\_72-cfg.dot) (ufunc\_outer)  -> 514(py\_test\_regression\_78-cfg.dot) (test\_mem\_axis\_minimization)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArray\_DescrFromType | NAX | Array API | | PyArray\_CanCastSafely | NAX | Array API |   Data/control flow direction:   * Python->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 foreign function  Location: if/branching statement |

| **Case 33** <https://github.com/numpy/numpy/commit/6d8010c4654cbc7e9a04495a4401548fa47e332c> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/scipy/scipy/pull/16649 , which points to an earlier report at <https://github.com/scipy/scipy/pull/12473> where it says    Thus, the symptom is incorrect output/result. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/numpy/numpy/commit/6d8010c4654cbc7e9a04495a4401548fa47e332c>    Bug is at line 6940 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  6940(c\_arrayobject\_220-cfg.dot) (\_array\_small\_type)  ...  -> 6919(c\_arrayobject\_220-cfg.dot) (\_array\_small\_type)  -> 5484(c\_multiarraymodule\_99-cfg.dot) (PyArray\_EquivTypes)  ...  -> 5476(c\_multiarraymodule\_99-cfg.dot) (PyArray\_EquivTypes)  -> 6919(c\_arrayobject\_220-cfg.dot) (\_array\_small\_type)  -> 6913(c\_arrayobject\_220-cfg.dot) (\_array\_small\_type)  -> 7171(c\_arrayobject\_223-cfg.dot) (\_array\_find\_type)  ...  -> 7029(c\_arrayobject\_223-cfg.dot) (\_array\_find\_type)  -> 8529(c\_arrayobject\_245-cfg.dot) (PyArray\_ObjectType)  ...  -> 8520(c\_arrayobject\_245-cfg.dot) (PyArray\_ObjectType)  -> 738(c\_\_dotblas\_9-cfg.dot) (dotblas\_innerproduct)  ...  -> 714(c\_\_dotblas\_9-cfg.dot) (dotblas\_innerproduct)  -> 435(py\_test\_ma\_23-cfg.dot) (check\_testTakeTransposeInnerOuter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_INCREF | Foreign | Reference Counting | | PyTypeNum\_ISEXTENDED | NAX | Array API | | PyArray\_DescrFromType | NAX | Array API | | PyArray\_CanCastSafely | NAX | Array API | | PyArray\_EquivTypes | NAX | Array API |   Data/control flow direction:   * Python->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/scipy/scipy/commit/0aee4a24bedb154d07a3b5ba016a385b1ac2910c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/8822> 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/scipy/scipy/commit/0aee4a24bedb154d07a3b5ba016a385b1ac2910c>    Bug is at line 603 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  603(c\_\_odepackmodule\_8-cfg.dot) (odepack\_odeint)  ...  -> 477(c\_\_odepackmodule\_8-cfg.dot) (odepack\_odeint)  -> 673(py\_test\_integrate\_160-cfg.dot) (test\_repeated\_t\_values)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_Check | Foreign | Tuple Objects | | Py\_INCREF | Foreign | Reference Counting | | PyArray\_Size | NAX | Array API | | PyArray\_DATA | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_ContiguousFromObject | NAX | Array API |   Data/control flow direction:   * Python->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 loop  Location: loop statement + function call |

| **Case 35** <https://github.com/scipy/scipy/commit/89acad947b34e3e906072adf383b8021935f4d3d> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/scipy/scipy/pull/10935where 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/scipy/scipy/commit/89acad947b34e3e906072adf383b8021935f4d3d>    Bug is at line 274 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  274(c\_ni\_filters\_2-cfg.dot) (NI\_Correlate)  ...  -> 159(c\_ni\_filters\_2-cfg.dot) (NI\_Correlate)  -> 230(c\_nd\_image\_9-cfg.dot) (Py\_Correlate)  …  -> 212(c\_nd\_image\_9-cfg.dot) (Py\_Correlate)  -> 1060(py\_test\_ndimage\_94-cfg.dot) (test\_generic\_filter01)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArray\_Size | NAX | Array API | | PyArray\_DATA | NAX | Array API | | PyArray\_DIMS | NAX | Array API |   Data/control flow direction:   * Python->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: Add a function to handle the data conversion  Location: function call |

| **Case 35** <https://github.com/scipy/scipy/commit/89acad947b34e3e906072adf383b8021935f4d3d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/10935> 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/scipy/scipy/commit/89acad947b34e3e906072adf383b8021935f4d3d>    Bug is at line 274 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  274(c\_ni\_filters\_2-cfg.dot) (NI\_Correlate)  ...  -> 159(c\_ni\_filters\_2-cfg.dot) (NI\_Correlate)  -> 230(c\_nd\_image\_9-cfg.dot) (Py\_Correlate)  …  -> 212(c\_nd\_image\_9-cfg.dot) (Py\_Correlate)  -> 1060(py\_test\_ndimage\_94-cfg.dot) (test\_generic\_filter01)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArray\_Size | NAX | Array API | | PyArray\_DATA | NAX | Array API | | PyArray\_DIMS | NAX | Array API |   Data/control flow direction:   * Python->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: Add a function to handle the data conversion  Location: function call |

| **Case 36** <https://github.com/scipy/scipy/commit/2a2d8be5f52481938c817eb7150fe7152958853d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/issues/742> 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/scipy/scipy/commit/2a2d8be5f52481938c817eb7150fe7152958853d>    Bug is at line 135 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  135(c\_ni\_measure\_1-cfg.dot) (NI\_Label)  ...  -> 49(c\_ni\_measure\_1-cfg.dot) (NI\_Label)  -> 693(c\_nd\_image\_24-cfg.dot) (Py\_Label)  ...  -> 682(c\_nd\_image\_24-cfg.dot) (Py\_Label)  -> 157(py\_measurements\_1-cfg.dot) (label)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:   * Python->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 37** <https://github.com/scipy/scipy/commit/60dc9730d5652f0632cd43caef437f01a734e374> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/13095> 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/scipy/scipy/commit/60dc9730d5652f0632cd43caef437f01a734e374>    Bug is at line 763 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  Path:  820(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  -> 818(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  -> 599(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 594(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 593(py\_interpolation\_6-cfg.dot) (affine\_transform)  -> 591(py\_interpolation\_6-cfg.dot) (affine\_transform)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Numerical and algorithmic errors |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Introducing a new parameter.  Location: if/branching statement |

| **Case 38** <https://github.com/scipy/scipy/commit/8fa4b7364d98659dd8fe28727f60d99a14b95850> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/scipy/scipy/pull/13139> 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/scipy/scipy/commit/8fa4b7364d98659dd8fe28727f60d99a14b95850>    Bug is at line 251 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  251(c\_ni\_interpolation\_3-cfg.dot) (\_get\_spline\_boundary\_mode)  ...  -> 245(c\_ni\_interpolation\_3-cfg.dot) (\_get\_spline\_boundary\_mode)  -> 772(c\_ni\_interpolation\_5-cfg.dot) (NI\_ZoomShift)  ...  -> 653(c\_ni\_interpolation\_5-cfg.dot) (NI\_ZoomShift)  -> 833(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  ...  -> 818(c\_nd\_image\_23-cfg.dot) (Py\_ZoomShift)  -> 675(py\_interpolation\_7-cfg.dot) (shift)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArray\_DATA | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | NPY\_UNLIKELY | NAX | System configuration |   Data/control flow direction:   * Python->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 the structure of if branch  Location: assignment statement + if/branching statement |

| **Case 39** <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/16649> , which points to an earlier report at <https://github.com/scipy/scipy/pull/12473> where it says    Thus, the symptom is segmentation fault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a>    Bug is at line 194 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  194(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 193(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 192(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 185(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 184(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 183(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 182(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 181(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 180(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 176(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 174(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 173(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 172(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 171(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 166(c\_\_fitpackmodule\_14-cfg.dot) (fitpack\_bispev)  -> 1052(py\_\_fitpack\_impl\_14-cfg.dot) (bisplev)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DATA | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_ContiguousFromObject | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type and add exception handling  Location: assignment statement |

| **Case 41** <https://github.com/scipy/scipy/commit/e2091e9e4be2dffed8c6535547e8519dc002fa74> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/12188> where it says    Thus, the symptom is segmentation fault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/e2091e9e4be2dffed8c6535547e8519dc002fa74>    Bug is at line 194 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  502(c\_\_fitpackmodule\_16-cfg.dot) (fitpack\_parcur)  -> 499(c\_\_fitpackmodule\_16-cfg.dot) (fitpack\_parcur)  ...  -> 413(c\_\_fitpackmodule\_16-cfg.dot) (fitpack\_parcur)  -> 280(py\_\_fitpack\_impl\_2-cfg.dot) (splprep)  -> 494(py\_interface\_25-cfg.dot) (transpose)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_SimpleNew | NAX | Array API | | PyArray\_ContiguousFromObject | NAX | Array API |   Data/control flow direction:   * Python->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 the conditional expression  Location: if/branching statement |

| **Case 42** <https://github.com/scipy/scipy/commit/3f4396e7c8e21ea0c432c1747b221e64c162b6f5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/12703> where it says    Thus, the symptom is invalid memory accesses |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/3f4396e7c8e21ea0c432c1747b221e64c162b6f5>    Bug is at line 261 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  261(c\_splinemodule\_15-cfg.dot) (FIRsepsym2d)  …  -> 235(c\_splinemodule\_15-cfg.dot) (FIRsepsym2d)  -> 257(py\_test\_bsplines\_27-cfg.dot) (test\_sepfir2d\_invalid\_filter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_MIN | NAX | Array API | | PyArray\_FromObject | NAX | Array API | | PyArray\_ObjectType | NAX | Array API |   Data/control flow direction:   * Python->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 if statement to check special case  Location: if/branching statement + NAX function |

| **Case 43** <https://github.com/scipy/scipy/commit/77b133df433a7e37e01023a88132d7ba47faac77> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/3534>, which points to an earlier report at <https://github.com/scipy/scipy/issues/3306> where it says    Thus, the symptom is value error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/77b133df433a7e37e01023a88132d7ba47faac77>    Bug is at line 339 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  339(c\_\_superluobject\_5-cfg.dot) (NRFormat\_from\_spMatrix)  -> 333(c\_\_superluobject\_5-cfg.dot) (NRFormat\_from\_spMatrix)  -> 331(c\_\_superluobject\_5-cfg.dot) (NRFormat\_from\_spMatrix)  -> 327(c\_\_superluobject\_5-cfg.dot) (NRFormat\_from\_spMatrix)  -> 148(c\_\_superlumodule\_6-cfg.dot) (Py\_gssv)  ...  -> 74(c\_\_superlumodule\_6-cfg.dot) (Py\_gssv)  -> 142(py\_linsolve\_5-cfg.dot) (spsolve)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArray\_ISCARRAY | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_EquivTypenums | NAX | Array API | | PyArray\_DESCR | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  NAX function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Change NAX function  Location: NAX function + conditional expression |

| **Case 44** <https://github.com/scipy/scipy/commit/88d625fc451d3714e640bbec693bb31abb6a05f5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/4143>, which points to an earlier report at <https://github.com/scipy/scipy/issues/3733> where it says    Thus, the symptom is segment fault |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/88d625fc451d3714e640bbec693bb31abb6a05f5>    Bug is at line 658 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  644(c\_ni\_filters\_6-cfg.dot) (NI\_Select)  ...  -> 642(c\_ni\_filters\_6-cfg.dot) (NI\_Select)  -> 769(c\_ni\_filters\_7-cfg.dot) (NI\_RankFilter)  ...  -> 253(c\_nd\_image\_13-cfg.dot) (Py\_RankFilter)  -> 1020(py\_filters\_25-cfg.dot) (\_rank\_filter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the data type  Location: variable initial statement |

| **Case 45** <https://github.com/scipy/scipy/commit/e16b190160fb15bb9bf590236c9e739c360ab320> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/4006> , which points to an earlier report at <https://github.com/scipy/scipy/pull/4003> where it says    Thus, the symptom is runtime warning |
| **Bug location:**  The issue was related to the placement of variable declarations in the SciPy codebase. The specific problem highlighted in the pull request discussion was an error caused by mixing declarations and code, which is not allowed in ISO C90 standard. In the original code, variable declarations were placed after executable statements, which violated the C90 standard requirement that all variable declarations must precede executable statements within a block. This led to a compilation error when building the code with a compiler enforcing C90 standards.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/e16b190160fb15bb9bf590236c9e739c360ab320>    Bug is at line 385 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  383(c\_ni\_filters\_4-cfg.dot) (NI\_MinOrMaxFilter1D)  …  -> 374(c\_ni\_filters\_4-cfg.dot) (NI\_MinOrMaxFilter1D)  -> 214(c\_nd\_image\_11-cfg.dot) (Py\_MinOrMaxFilter1D)  …  -> 201(c\_nd\_image\_11-cfg.dot) (Py\_MinOrMaxFilter1D)  -> 815(py\_filters\_20-cfg.dot) (minimum\_filter1d)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: Move declarations before executable statemen  Location: variable initialization statement |

| **Case 46** <https://github.com/tskit-dev/tskit/commit/568a9a953d549ae9a48f0f6d98435249841b7b5e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/tskit-dev/tskit/issues/453> where it says    Thus, the symptom is library error |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/tskit-dev/tskit/commit/568a9a953d549ae9a48f0f6d98435249841b7b5e>    Bug is at line 9041 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  9041(c\_\_tskitmodule\_336-cfg.dot) (Tree\_get\_newick)  …  -> 9016(c\_\_tskitmodule\_336-cfg.dot) (Tree\_get\_newick)  -> 1803(py\_test\_lowlevel\_329-cfg.dot) (test\_newick\_precision)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * Python->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 the variable to a larger number  Location: assignment statement |

| **Case 47** <https://github.com/scipy/scipy/commit/5910e04cb6b58d795623d3d4e30369e733ef41a0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/issues/7395> where it says    Thus, the symptom is segment fault |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/scipy/scipy/commit/5910e04cb6b58d795623d3d4e30369e733ef41a0>    Bug is at line 502 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  502(c\_\_fitpackmodule\_16-cfg.dot) (fitpack\_parcur)  …  -> 413(c\_\_fitpackmodule\_16-cfg.dot) (fitpack\_parcur)  -> 280(py\_\_fitpack\_impl\_2-cfg.dot) (splprep)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_SimpleNew | NAX | Array API | | PyArray\_ContiguousFromObject | NAX | Array API |   Data/control flow direction:   * Python->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 48** <https://github.com/scipy/scipy/commit/ed220ccdf4bab8fd25293afec26a8cb321f12921> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/scipy/scipy/pull/14327> , which points to an earlier report at <https://github.com/scipy/scipy/issues/14273> 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/scipy/scipy/commit/ed220ccdf4bab8fd25293afec26a8cb321f12921>    Bug is at line 903 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  903(c\_sigtoolsmodule\_27-cfg.dot) (PyArray\_OrderFilterND)  ...  -> 848(c\_sigtoolsmodule\_27-cfg.dot) (PyArray\_OrderFilterND)  -> 1210(c\_sigtoolsmodule\_30-cfg.dot) (sigtools\_order\_filterND)  ...  -> 1203(c\_sigtoolsmodule\_30-cfg.dot) (sigtools\_order\_filterND)  -> 1475(py\_signaltools\_22-cfg.dot) (order\_filter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArray\_TYPE | NAX | Array API | | PyArray\_SimpleNew | NAX | Array API | | PyArray\_DIMS | NAX | Array API | | PyArray\_NDIM | NAX | Array API | | PyArray\_ITEMSIZE | NAX | Array API | | PyArray\_Zero | NAX | Array API | | PyArray\_ContiguousFromObject | NAX | Array API | | PyArray\_ObjectType | NAX | Array API |   Data/control flow direction:   * Python->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: Boundary check against value range/limit  Location: conditional expression + if/branching statement |

| **Case 49** <https://github.com/CheetahTemplate3/cheetah3/commit/31574d601ed6b675eb2fe8dc82659864bd3ace14> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/CheetahTemplate3/cheetah3/issues/26> where it says    Thus, the symptom is segment fault |
| **Bug location:**  Three users of `checkForNameInNameSpaceAndReturnIfFound` in \_namemapper.c were not checking to see that `numChunks > 0`, and so were using `nameChunks[0]`, which is uninitialized if the name passed in is the empty string.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/CheetahTemplate3/cheetah3/commit/31574d601ed6b675eb2fe8dc82659864bd3ace14>    Bug is at line 271 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  271(c\_\_namemapper\_9-cfg.dot) (namemapper\_valueForName)  ...  -> 249(c\_\_namemapper\_9-cfg.dot) (namemapper\_valueForName)  -> 112(py\_NameMapper\_25-cfg.dot) (VFN)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyNamemapper\_valueForName | NAX | Mapper API |   Data/control flow direction:   * Python->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: NAX function + assignment statement |

| **Case 50** [https://github.com/beancount/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213](https://github.com/sjfourie2/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213) |
| --- |
| **Symptoms:**  Directly associated issue report: [https://github.com/beancount/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213](https://github.com/sjfourie2/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213) where it says    Thus, the symptom is segment fault |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  [https://github.com/beancount/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213](https://github.com/sjfourie2/beancount/commit/72fa13564b89a24ca28d6a3ba96c6a696783e213)    Bug is at line 238 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  238(c\_parser\_10-cfg.dot) (lexer\_next)  ...  -> 209(c\_parser\_10-cfg.dot) (lexer\_next)  -> 230(py\_lexer\_28-cfg.dot) (lex\_iter)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: Force casting the type of an argument in a foreign function  Location: Foreign function |

| **Case 51** <https://github.com/cjdrake/pyeda/commit/cb56a584b133e6062cd1efcedf36a7603bd82c6a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/cjdrake/pyeda/issues/116> where it says    Thus, the symptom is memory leak |
| **Bug location:**  The ExprNode\_restrict and ExprNode\_compose functions do not currently delete the dictionary used during restrict/compose operations.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/cjdrake/pyeda/commit/cb56a584b133e6062cd1efcedf36a7603bd82c6a>      Bug is at line 522, 570 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  522(c\_exprnodemodule\_26-cfg.dot) (ExprNode\_restrict)  ...  -> 497(c\_exprnodemodule\_26-cfg.dot) (ExprNode\_restrict)  -> 1302(py\_expr\_277-cfg.dot) (\_iter\_backtrack)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | … | NAX |  |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + type conversion  Location: if/branching statement + type-conversion expression assignment |

| **Case 52** <https://github.com/univention/univention-corporate-server/commit/ac3d052b9837220bd8ed05800e0aa2ecbb998584> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/univention/univention-corporate-server/commit/ac3d052b9837220bd8ed05800e0aa2ecbb998584> 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/univention/univention-corporate-server/commit/ac3d052b9837220bd8ed05800e0aa2ecbb998584>    Bug is at line 102 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  102(c\_creds\_3-cfg.dot) (creds\_new)  ...  -> 79(c\_creds\_3-cfg.dot) (creds\_new)  -> 115(py\_test\_4-cfg.dot) (test\_creds)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyObject\_NEW | Foreign | Allocating Objects on the Heap | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 statement  Location: After foreign function |

| **Case 53** <https://github.com/python-pillow/Pillow/commit/6f73691564908f70967cf76b8f4f88e390f57fe9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/7257> , which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/7245> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  The primary purpose of this function seems to be to apply a mask (imMask) to another image (imOut). It operates differently based on the mode of imOut and whether it has an 8-bit representation (image8) or a more complex pixel structure. The masking process involves blending pixels from imOut with values from imMask, modulated by the ink color.  The modification highlighted in your code snippet (\*out = BLEND(\*mask, \*out, ink[1], tmp1);) is an alteration within a conditional block that checks if the mode of imOut is "I;16". This suggests that the image is in a 16-bit integer format. The original line was blending using only the first component of ink (ink[0]), but the modification changes it to use the second component (ink[1]).  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/6f73691564908f70967cf76b8f4f88e390f57fe9>    Bug is at line 428 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  428(c\_Paste\_9-cfg.dot) (fill\_mask\_L)  ...  -> 400(c\_Paste\_9-cfg.dot) (fill\_mask\_L)  -> 609(c\_Paste\_12-cfg.dot) (ImagingFill2)  ...  -> 549(c\_Paste\_12-cfg.dot) (ImagingFill2)  -> 1402(c\_\_imaging\_49-cfg.dot) (\_paste)  ...  -> 1375(c\_\_imaging\_49-cfg.dot) (\_paste)  -> 46(py\_test\_image\_resample\_4-cfg.dot) (test\_modify\_after\_resizing)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | strncmp | Foreign | String conversion and formatting | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingError\_Mismatch | NAX | Exception Handling |   Data/control flow direction:   * Python->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: assignment statement + function call |

| **Case 54** <https://github.com/python-pillow/Pillow/commit/eb89e3a2b1833eb12124fb5f1f26f78342286143> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/6874> where it says    Thus, the symptom is segment fault. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/eb89e3a2b1833eb12124fb5f1f26f78342286143>    Bug is at line 266 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  241(c\_BoxBlur\_4-cfg.dot) (ImagingBoxBlur)  -> 237(c\_BoxBlur\_4-cfg.dot) (ImagingBoxBlur)  -> 232(c\_BoxBlur\_4-cfg.dot) (ImagingBoxBlur)  -> 2106(c\_\_imaging\_65-cfg.dot) (\_box\_blur)  …  -> 2089(c\_\_imaging\_65-cfg.dot) (\_box\_blur)  -> 25(py\_test\_box\_blur\_2-cfg.dot) (box\_blur)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingError\_ValueError | NAX | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Error handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a if statement  Location: if/branching statement + error handling NAX function |

| **Case 55** <https://github.com/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/6429> , which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/6428> 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/scipy/scipy/commit/a13a7b82d4664c767575097c4a60c0b5fcd1098a>    Bug is at line 98 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  98(py\_PSDraw\_7-cfg.dot) (rectangle)  -> 197(py\_PdfParser\_33-cfg.dot) (write)  -> 165(py\_PdfParser\_33-cfg.dot) (write)  -> 164(py\_PdfParser\_33-cfg.dot) (write)  -> 163(py\_PdfParser\_33-cfg.dot) (write)  -> 164(py\_WebPImagePlugin\_10-cfg.dot) (tell)  -> 163(py\_WebPImagePlugin\_10-cfg.dot) (tell)  -> 162(py\_WebPImagePlugin\_10-cfg.dot) (tell)  -> 676(py\_test\_file\_gif\_47-cfg.dot) (test\_no\_transparency\_in\_second\_frame)  -> 674(py\_test\_file\_gif\_47-cfg.dot) (test\_no\_transparency\_in\_second\_frame)  -> 680(py\_test\_file\_gif\_47-cfg.dot) (test\_no\_transparency\_in\_second\_frame)  -> 1235(c\_\_imaging\_45-cfg.dot) (\_histogram)  -> 1234(c\_\_imaging\_45-cfg.dot) (\_histogram)  -> 1233(c\_\_imaging\_45-cfg.dot) (\_histogram)  -> 1232(c\_\_imaging\_45-cfg.dot) (\_histogram)  -> 1224(c\_\_imaging\_45-cfg.dot) (\_histogram)  -> 680(py\_test\_file\_gif\_47-cfg.dot) (test\_no\_transparency\_in\_second\_frame)  Along the backward search path until origin, the cross-language functions involved are:       | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C ->Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function call misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument in the function call  Location: function call |

| **Case 56** <https://github.com/python-pillow/Pillow/commit/2e5ce839babe540865445316e7abf1cecef97264> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5633> , which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/5632> where it says    Thus, the symptom is error message (attribute error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/2e5ce839babe540865445316e7abf1cecef97264>    Bug is at line 2183 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2183(py\_Image\_93-cfg.dot) (save)  -> 6(py\_\_util\_1-cfg.dot) (isPath)  -> 5(py\_\_util\_1-cfg.dot) (isPath)  -> 2183(py\_Image\_93-cfg.dot) (save)  -> 2182(py\_Image\_93-cfg.dot) (save)  -> 2181(py\_Image\_93-cfg.dot) (save)  -> 2151(py\_Image\_93-cfg.dot) (save)  -> 135(py\_test\_file\_webp\_17-cfg.dot) (test\_write\_encoding\_error\_message)  -> 134(py\_test\_file\_webp\_17-cfg.dot) (test\_write\_encoding\_error\_message)  -> 133(py\_test\_file\_webp\_17-cfg.dot) (test\_write\_encoding\_error\_message)  -> 352(py\_PyAccess\_106-cfg.dot) (new)  -> 351(py\_PyAccess\_106-cfg.dot) (new)  -> 350(py\_PyAccess\_106-cfg.dot) (new)  -> 349(py\_PyAccess\_106-cfg.dot) (new)  -> 348(py\_PyAccess\_106-cfg.dot) (new)  -> 133(py\_test\_file\_webp\_17-cfg.dot) (test\_write\_encoding\_error\_message)  -> 573(py\_Image\_35-cfg.dot) (\_new)  -> 572(py\_Image\_35-cfg.dot) (\_new)  -> 1153(py\_Image\_59-cfg.dot) (copy)  -> 977(c\_\_imaging\_34-cfg.dot) (\_copy)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyImagingNew | NAX | Image API |   Data/control flow direction:   * C-> Python |
| **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 execution order of branches in the if statement.  Location: if/branching statement |

| **Case 57** <https://github.com/python-pillow/Pillow/commit/5f39e8e60ae0662b850484b2ad7a1eaa83677dd3> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5613> , which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/5612> 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/python-pillow/Pillow/commit/5f39e8e60ae0662b850484b2ad7a1eaa83677dd3>    Bug is at line 182 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  182(py\_PsdImagePlugin\_15-cfg.dot) (\_layerinfo)  -> 828(py\_test\_file\_jpeg\_68-cfg.dot) (read)  -> 827(py\_test\_file\_jpeg\_68-cfg.dot) (read)  -> 2340(py\_Image\_98-cfg.dot) (tell)  -> 2331(py\_Image\_98-cfg.dot) (tell)  -> 45(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 42(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 140(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 139(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 310(py\_ImageFile\_11-cfg.dot) (\_seek\_check)  -> 300(py\_ImageFile\_11-cfg.dot) (\_seek\_check)  -> 298(py\_ImageFile\_11-cfg.dot) (\_seek\_check)  -> 139(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 138(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 41(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 38(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 37(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 70(py\_WalImageFile\_1-cfg.dot) (open)  -> 67(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 64(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 63(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 2317(c\_\_imaging\_75-cfg.dot) (\_split)  -> 2316(c\_\_imaging\_75-cfg.dot) (\_split)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingError\_ModeError | NAX | Exception Handling |   Data/control flow direction:   * C-> Python |
| **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: Changed the originally called function to another one.  Location: assignment statement |

| **Case 58** <https://github.com/python-pillow/Pillow/commit/d3a3b6dddba4af1153aa6c01be6d3be5f43636ca> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5408> 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/python-pillow/Pillow/commit/d3a3b6dddba4af1153aa6c01be6d3be5f43636ca>    Bug is at line 143 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  143(py\_JpegImagePlugin\_2-cfg.dot) (APP)  ...  -> 64(py\_JpegImagePlugin\_2-cfg.dot) (APP)  -> 811(py\_test\_file\_jpeg\_67-cfg.dot) (read)  -> 810(py\_test\_file\_jpeg\_67-cfg.dot) (read)  -> 302(py\_ImImagePlugin\_7-cfg.dot) (tell)  ...  -> 301(py\_ImImagePlugin\_7-cfg.dot) (tell)  -> 51(py\_test\_imagesequence\_4-cfg.dot) (\_test\_multipage\_tiff)  -> 50(py\_test\_imagesequence\_4-cfg.dot) (\_test\_multipage\_tiff)  -> 46(py\_test\_file\_png\_2-cfg.dot) (load)  -> 73(py\_WalImageFile\_1-cfg.dot) (open)  ...  -> 70(py\_WalImageFile\_1-cfg.dot) (open)  -> 67(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 64(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 63(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 2308(c\_\_imaging\_75-cfg.dot) (\_split)  -> 2307(c\_\_imaging\_75-cfg.dot) (\_split)  -> 75(c\_Bands\_2-cfg.dot) (ImagingSplit)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingSplit | NAX | Image API |   Data/control flow direction:   * C-> Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  index error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Changed the index number in a array variable  Location: assignment statement |

| **Case 59** <https://github.com/python-pillow/Pillow/commit/d3a3b6dddba4af1153aa6c01be6d3be5f43636ca> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5400> 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/python-pillow/Pillow/commit/d3a3b6dddba4af1153aa6c01be6d3be5f43636ca>    Bug is at line 116 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  116(py\_SgiImagePlugin\_3-cfg.dot) (\_open)  -> 115(py\_SgiImagePlugin\_3-cfg.dot) (\_open)  -> 114(py\_SgiImagePlugin\_3-cfg.dot) (\_open)  -> 51(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 50(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 49(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 140(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 139(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 310(py\_ImageFile\_11-cfg.dot) (\_seek\_check)  -> 302(py\_ImImagePlugin\_7-cfg.dot) (tell)  -> 301(py\_ImImagePlugin\_7-cfg.dot) (tell)  -> 45(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  ...  -> 37(py\_test\_file\_container\_4-cfg.dot) (test\_seek\_mode\_1)  -> 70(py\_WalImageFile\_1-cfg.dot) (open)  -> 67(py\_WalImageFile\_2-cfg.dot) (imopen)  ...  -> 63(py\_WalImageFile\_2-cfg.dot) (imopen)  -> 2308(c\_\_imaging\_75-cfg.dot) (\_split)  -> 2307(c\_\_imaging\_75-cfg.dot) (\_split)  -> 75(c\_Bands\_2-cfg.dot) (ImagingSplit)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingSplit | NAX | Image API |   Data/control flow direction:   * C-> Python |
| **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: Changed the argument error  Location: assignment statement |

| **Case 60** <https://github.com/python-pillow/Pillow/commit/9a683db33945aacf2fa9a277f680e29d54685930> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5274> 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/python-pillow/Pillow/commit/9a683db33945aacf2fa9a277f680e29d54685930>    Bug is at line 80 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  79(c\_Fill\_2-cfg.dot) (ImagingFillLinearGradient)  ...  -> 65(c\_Fill\_2-cfg.dot) (ImagingFillLinearGradient)  ...  -> 71(c\_Fill\_2-cfg.dot) (ImagingFillLinearGradient)  -> 40(c\_Except\_3-cfg.dot) (ImagingError\_ModeError)  -> 290(c\_\_imaging\_14-cfg.dot) (ImagingError\_ValueError)  ...  -> 286(c\_\_imaging\_14-cfg.dot) (ImagingError\_ValueError)  -> 1152(c\_Geometry\_30-cfg.dot) (ImagingTransform)  -> 1140(c\_Geometry\_30-cfg.dot) (ImagingTransform)  -> 1126(c\_Geometry\_30-cfg.dot) (ImagingTransform)  -> 1963(c\_\_imaging\_62-cfg.dot) (\_transform2)  ...  -> 1914(c\_\_imaging\_62-cfg.dot) (\_transform2)  -> 2532(py\_Image\_98-cfg.dot) (\_\_transformer)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingError\_ModeError | NAX | Exception Handling | | ImagingNewDirty | NAX | Image API |   Data/control flow direction:   * Python->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 + other cases handling  Location: if/branching statement |

| **Case 61** <https://github.com/python-pillow/Pillow/commit/96f2ef14265b3515c17a64e7ca0d5fe7bd42cd05> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5274> 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/python-pillow/Pillow/commit/96f2ef14265b3515c17a64e7ca0d5fe7bd42cd05>    Bug is at line 177 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  177(py\_test\_image\_reduce\_8-cfg.dot) (assert\_compare\_images)  -> 178(py\_test\_image\_reduce\_8-cfg.dot) (assert\_compare\_images)  -> 174(py\_test\_image\_reduce\_8-cfg.dot) (assert\_compare\_images)  -> 51(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 50(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 49(py\_ImageSequence\_5-cfg.dot) (\_\_next\_\_)  -> 137(py\_PsdImagePlugin\_4-cfg.dot) (seek)  ...  -> 135(py\_PsdImagePlugin\_4-cfg.dot) (seek)  -> 125(py\_PcfFontFile\_9-cfg.dot) (\_load\_properties)  -> 124(py\_PcfFontFile\_9-cfg.dot) (\_load\_properties)  -> 122(py\_PcfFontFile\_9-cfg.dot) (\_load\_properties)  -> 605(py\_TiffImagePlugin\_39-cfg.dot) (\_\_iter\_\_)  -> 604(py\_TiffImagePlugin\_39-cfg.dot) (\_\_iter\_\_)  -> 19(py\_test\_image\_resize\_3-cfg.dot) (test\_nearest\_mode)  ...  -> 32(py\_test\_image\_resize\_3-cfg.dot) (test\_nearest\_mode)  -> 1793(c\_\_imaging\_59-cfg.dot) (\_resize)  -> 174(c\_\_imaging\_3-cfg.dot) (PyImagingNew)  -> 173(c\_\_imaging\_3-cfg.dot) (PyImagingNew)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyObject\_New | Foreign | Allocating Objects on the Heap | | ImagingDelete | NAX | Image API | | PyImagingNew | NAX | Image API |   Data/control flow direction:   * C-> Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Broken string |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the string  Location: assignment statement |

| **Case 62** <https://github.com/python-pillow/Pillow/commit/a69c37738ab5bf370e007f762272710a86a9bd5a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/5274> 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/python-pillow/Pillow/commit/a69c37738ab5bf370e007f762272710a86a9bd5a>    Bug is at line 187 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  187(py\_ImageTk\_9-cfg.dot) (paste)  -> 42(c\_\_imagingtk\_3-cfg.dot) (\_tkinit)  -> 41(c\_\_imagingtk\_3-cfg.dot) (\_tkinit)  -> 34(c\_\_imagingtk\_3-cfg.dot) (\_tkinit)  -> 187(py\_ImageTk\_9-cfg.dot) (paste)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C-> Python |
| **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: add a if statement + change called function  Location: function call |

| **Case 63** <https://github.com/python-pillow/Pillow/commit/bfa80cd3b387f5f6c1bce12172803a8b7f7c78e2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/2296>, which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/2268> 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/python-pillow/Pillow/commit/bfa80cd3b387f5f6c1bce12172803a8b7f7c78e2>    Bug is at line 35 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  35(c\_Access\_2-cfg.dot) (add\_item)  -> 33(c\_Access\_2-cfg.dot) (add\_item)  -> 27(c\_Access\_1-cfg.dot) (hash)  -> 25(c\_Access\_1-cfg.dot) (hash)  -> 24(c\_Access\_1-cfg.dot) (hash)  -> 21(c\_Access\_1-cfg.dot) (hash)  ...  -> 21(c\_Access\_1-cfg.dot) (hash)  -> 28(py\_make\_hash\_2-cfg.dot) (check)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 64** <https://github.com/python-pillow/Pillow/commit/ca3f6a25f40f6c51d69c71f1206e337684a36ef8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/issues/1902> where it says    Thus, the symptom is error message (overflow error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/ca3f6a25f40f6c51d69c71f1206e337684a36ef8>    Bug is at line 45 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  41(c\_\_imagingtk\_3-cfg.dot) (\_tkinit)  -> 34(c\_\_imagingtk\_3-cfg.dot) (\_tkinit)  -> 187(py\_ImageTk\_9-cfg.dot) (paste)  -> 185(py\_ImageTk\_9-cfg.dot) (paste)  -> 181(py\_ImageTk\_9-cfg.dot) (paste)  -> 133(py\_PngImagePlugin\_7-cfg.dot) (call)  -> 136(c\_decode\_5-cfg.dot) (\_decode)  -> 132(c\_decode\_5-cfg.dot) (\_decode)  -> 130(c\_decode\_5-cfg.dot) (\_decode)  -> 124(c\_decode\_5-cfg.dot) (\_decode)  -> 123(c\_decode\_5-cfg.dot) (\_decode)  -> 116(c\_decode\_5-cfg.dot) (\_decode)  -> 133(py\_PngImagePlugin\_7-cfg.dot) (call)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 65** <https://github.com/python-pillow/Pillow/commit/7e484eba77d1fe3f62d9a91f90bf508d03b4ea1c> |
| --- |
| **Symptoms:**  No description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/7e484eba77d1fe3f62d9a91f90bf508d03b4ea1c>    Bug is at line 2215, 2233 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2215(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2210(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2209(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2208(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2206(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2203(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2202(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2199(c\_\_imaging\_85-cfg.dot) (\_font\_text\_asBytes)  -> 2315(c\_\_imaging\_87-cfg.dot) (\_font\_getsize)  -> 2312(c\_\_imaging\_87-cfg.dot) (\_font\_getsize)  -> 2305(c\_\_imaging\_87-cfg.dot) (\_font\_getsize)  -> 500(py\_test\_imagefont\_47-cfg.dot) (test\_imagefont\_getters)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyString\_AsStringAndSize | Foreign | String/Bytes Objects | | PyBytes\_AsStringAndSize | Foreign | Bytes Objects | | PyBytes\_Check | Foreign | Bytes Objects | | PyUnicode\_AsLatin1String | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * Python->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 one to the variables  Location: assignment statement + function call |

| **Case 66** <https://github.com/python-pillow/Pillow/commit/e71757aa6f64f698a16acaece9bc5557302a1fe8> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/2634> 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/python-pillow/Pillow/commit/e71757aa6f64f698a16acaece9bc5557302a1fe8>    Bug is at line 2249 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  -> 2249(c\_\_imaging\_87-cfg.dot) (\_font\_getmask)  …  -> 2225(c\_\_imaging\_87-cfg.dot) (\_font\_getmask)  -> 310(py\_test\_imagefont\_31-cfg.dot) (test\_rotated\_transposed\_font\_get\_mask)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: free the variables  Location: before return statement |

| **Case 67** <https://github.com/python-pillow/Pillow/commit/a0eaf06cc5f62a6fb6de556989ac1014ff3348ea> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/issues/254> 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/python-pillow/Pillow/commit/a0eaf06cc5f62a6fb6de556989ac1014ff3348ea>    Bug is at line 333 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  333(c\_Storage\_8-cfg.dot) (ImagingNewBlock)  ...  -> 315(c\_Storage\_8-cfg.dot) (ImagingNewBlock)  -> 373(c\_Storage\_9-cfg.dot) (ImagingNew)  ...  -> 358(c\_Storage\_9-cfg.dot) (ImagingNew)  -> 247(c\_Access\_21-cfg.dot) (ImagingAccessNew)  -> 246(c\_Access\_21-cfg.dot) (ImagingAccessNew)  -> 245(c\_Access\_21-cfg.dot) (ImagingAccessNew)  -> 20(py\_make\_hash\_1-cfg.dot) (hash)  -> 18(py\_make\_hash\_1-cfg.dot) (hash)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingNewPrologue | NAX | Image API |   Data/control flow direction:   * Python->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: memset the variable  Location: after using the variable |

| **Case 68** <https://github.com/python-pillow/Pillow/commit/6e2075e25d49fa0ffaec5892b64e3dd4cd39a9fd> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/pull/574>, which points to an earlier report at <https://github.com/python-pillow/Pillow/issues/510> 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/python-pillow/Pillow/commit/6e2075e25d49fa0ffaec5892b64e3dd4cd39a9fd>    Bug is at line 789 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  780(py\_Image\_42-cfg.dot) (quantize)  -> 779(py\_Image\_42-cfg.dot) (quantize)  -> 324(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 323(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 321(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 512(py\_Image\_31-cfg.dot) (\_dump)  -> 508(py\_Image\_31-cfg.dot) (\_dump)  -> 507(py\_Image\_31-cfg.dot) (\_dump)  -> 506(py\_Image\_31-cfg.dot) (\_dump)  -> 39(py\_test\_image\_3-cfg.dot) (test\_internals)  -> 38(py\_test\_image\_3-cfg.dot) (test\_internals)  -> 223(py\_tester\_20-cfg.dot) (tempfile)  -> 222(py\_tester\_20-cfg.dot) (tempfile)  -> 215(py\_tester\_20-cfg.dot) (tempfile)  -> 348(py\_TiffImagePlugin\_15-cfg.dot) (\_\_iter\_\_)  ...  -> 347(py\_TiffImagePlugin\_15-cfg.dot) (\_\_iter\_\_)  -> 151(py\_ImageOps\_5-cfg.dot) (colorize)  -> 150(py\_ImageOps\_5-cfg.dot) (colorize)  -> 149(py\_ImageOps\_5-cfg.dot) (colorize)  -> 42(py\_ImageOps\_2-cfg.dot) (\_color)  -> 39(py\_ImageOps\_2-cfg.dot) (\_color)  -> 10(py\_\_util\_3-cfg.dot) (isStringType)  -> 9(py\_\_util\_3-cfg.dot) (isStringType)  -> 145(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 144(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 138(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 137(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 211(py\_ImageDraw\_13-cfg.dot) (shape)  -> 210(py\_ImageDraw\_13-cfg.dot) (shape)  -> 97(c\_\_sane\_4-cfg.dot) (SaneDev\_close)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | ImagingNewPrologue | NAX | Image API |   Data/control flow direction:   * C-> Python |
| **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 + exception handling  Location: if/branching statement |

| **Case 69** <https://github.com/python-pillow/Pillow/commit/9552d9bc66d159f349863a9190e4ae882d870d6a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-pillow/Pillow/issues/510> where it says    Thus, the symptom is file handler leak |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/python-pillow/Pillow/commit/9552d9bc66d159f349863a9190e4ae882d870d6a>    Bug is at line 732 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  732(py\_TiffImagePlugin\_59-cfg.dot) (\_load\_libtiff)  ...  -> 684(py\_TiffImagePlugin\_59-cfg.dot) (\_load\_libtiff)  -> 324(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 323(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 321(py\_EpsImagePlugin\_20-cfg.dot) (load)  -> 512(py\_Image\_31-cfg.dot) (\_dump)  -> 508(py\_Image\_31-cfg.dot) (\_dump)  -> 507(py\_Image\_31-cfg.dot) (\_dump)  -> 506(py\_Image\_31-cfg.dot) (\_dump)  -> 39(py\_test\_image\_3-cfg.dot) (test\_internals)  -> 38(py\_test\_image\_3-cfg.dot) (test\_internals)  -> 223(py\_tester\_20-cfg.dot) (tempfile)  -> 222(py\_tester\_20-cfg.dot) (tempfile)  -> 215(py\_tester\_20-cfg.dot) (tempfile)  -> 348(py\_TiffImagePlugin\_15-cfg.dot) (\_\_iter\_\_)  -> 347(py\_TiffImagePlugin\_15-cfg.dot) (\_\_iter\_\_)  ...  -> 347(py\_TiffImagePlugin\_15-cfg.dot) (\_\_iter\_\_)  -> 151(py\_ImageOps\_5-cfg.dot) (colorize)  -> 150(py\_ImageOps\_5-cfg.dot) (colorize)  -> 149(py\_ImageOps\_5-cfg.dot) (colorize)  -> 42(py\_ImageOps\_2-cfg.dot) (\_color)  -> 39(py\_ImageOps\_2-cfg.dot) (\_color)  -> 10(py\_\_util\_3-cfg.dot) (isStringType)  -> 9(py\_\_util\_3-cfg.dot) (isStringType)  -> 145(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 144(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 138(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 137(py\_ImageDraw\_7-cfg.dot) (\_getink)  -> 211(py\_ImageDraw\_13-cfg.dot) (shape)  -> 210(py\_ImageDraw\_13-cfg.dot) (shape)  -> 97(c\_\_sane\_4-cfg.dot) (SaneDev\_close)    Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_INCREF | Foreign | Reference Counting | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C-> Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  File I/O error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: close the the file after using  Location: if/branching statement |

| **Case 70** <https://github.com/ultrajson/ultrajson/commit/335ac83c1308b928bb316eb302114d76543868cf> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ultrajson/ultrajson/pull/426> , which points to an earlier report at <https://github.com/ultrajson/ultrajson/issues/415> 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/ultrajson/ultrajson/commit/335ac83c1308b928bb316eb302114d76543868cf>    Bug is at line 786 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  786(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 755(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 756(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 772(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 753(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 752(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 751(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 750(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 749(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 748(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 747(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 745(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 741(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 739(c\_objToJSON\_39-cfg.dot) (objToJSON)  -> 306(py\_benchmark\_25-cfg.dot) (benchmark\_medium\_complex\_object)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 variable format in argument  Location: if/branching statement + foreign function |

| **Case 71** <https://github.com/ultrajson/ultrajson/commit/a920bfa9d85bcd78836b866d1be80c1e3dcca1da> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ultrajson/ultrajson/issues/429> where it says    Thus, the symptom is incorrect result/output. |
| **Bug location:**  The code change involves adjusting where newlines and indentation are added. Now, they are added between elements of a non-empty array or object, instead of between the start and end markers.  This means, if the array or object is empty (count == 0), there will be no additional newline and indentation.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/ultrajson/ultrajson/commit/a920bfa9d85bcd78836b866d1be80c1e3dcca1da>    Bug is at line 690 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  690(c\_ultrajsonenc\_11-cfg.dot) (encode)  ...  -> 593(c\_ultrajsonenc\_11-cfg.dot) (encode)  -> 66(py\_test\_ujson\_6-cfg.dot) (test\_double\_long\_issue)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Buffer\_AppendCharUnchecked | NAX | Buffer management | | Buffer\_EscapeStringUnvalidated | NAX | Buffer management | | Buffer\_Reserve | NAX | Buffer management |   Data/control flow direction:   * Python->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: NAX function |

| **Case 72** <https://github.com/pygame/pygame/commit/8568694cb569b8e4622a06f83cc2e00423da595c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/2959> 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/pygame/pygame/commit/8568694cb569b8e4622a06f83cc2e00423da595c>    Bug is at line 963 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  963(c\_image\_13-cfg.dot) (image\_fromstring)  -> 962(c\_image\_13-cfg.dot) (image\_fromstring)  -> 958(c\_image\_13-cfg.dot) (image\_fromstring)  -> 954(c\_image\_13-cfg.dot) (image\_fromstring)  -> 953(c\_image\_13-cfg.dot) (image\_fromstring)  -> 948(c\_image\_13-cfg.dot) (image\_fromstring)  -> 531(py\_image\_test\_25-cfg.dot) (test\_fromstring\_\_and\_tostring)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyBytes\_AsStringAndSize | Foreign | Bytes Objects | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  error handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Change the string  Location: function call |

| **Case 73** <https://github.com/pygame/pygame/commit/4b1e75e537a08ba90c10c96160d5c80e53e69b06> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/2406> , which points to an earlier report at <https://github.com/pygame/pygame/issues/2359> where it says    Thus, the symptom is incorrect result/output. |
| **Bug location:**  The bug seems to be related to how the function handled rectangles with very small widths or heights. In the original code, there was no specific handling for rectangles where either the width or height was less than 2 pixels. This could lead to unexpected behavior or incorrect rendering of such rectangles, especially when rounded corners (specified by the radius and other \*\_radius parameters) were involved.  The updated code introduces a check to handle these cases. If the rectangle's width or height is less than 2 pixels, or if there is no rounded corner (radius and all \*\_radius are non-positive), it falls back to a simpler rendering logic that either draws a polygon (if width > 0) or fills a rectangle using SDL\_FillRect.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/4b1e75e537a08ba90c10c96160d5c80e53e69b06>    Bug is at line 975 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  975(c\_draw\_21-cfg.dot) (rect)  …  -> 854(c\_draw\_21-cfg.dot) (rect)  -> 4169(py\_draw\_test\_331-cfg.dot) (test\_draw\_diamond)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 against value range/limit + add more caculation  Location: if/branching statement |

| **Case 74** <https://github.com/pygame/pygame/commit/e40d00db1f8015e8f37624f83a0bd334547cd8dc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/1580> where it says    Thus, the symptom is function Breakdown. |
| **Bug location:**  In the modified code, there's special handling for SDL\_KEYDOWN and SDL\_KEYUP event types. If the event is one of these two, the code now extracts the relevant keyboard event data (such as key, scancode, mod) from the Python event object and correctly populates these into the corresponding fields of the SDL\_Event structure.  This means that when you post a keyboard event through the Pygame interface, it can now be properly parsed and treated as a standard SDL keyboard event, rather than a generic user event.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/e40d00db1f8015e8f37624f83a0bd334547cd8dc>    Bug is at line 1835 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1835(c\_event\_30-cfg.dot) (pg\_event\_post)  …  -> 1815(c\_event\_30-cfg.dot) (pg\_event\_post)  -> 311(py\_event\_test\_31-cfg.dot) (test\_post\_large\_user\_event)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyDict\_GetItemString | Foreign | Dictionary Objects | | PyInt\_Check | Foreign | Plain Integer Objects | | PyLong\_AsLong | Foreign | Integer Objects |   Data/control flow direction:   * Python->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 more code to handle the special case  Location: if/branching statement |

| **Case 75** <https://github.com/pygame/pygame/commit/f3ac656da14e0a40335c54a6b453c2d15184d4a2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/2398> , which points to an earlier report at <https://github.com/pygame/pygame/issues/2347> where it says    Thus, the symptom is function lacking. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/f3ac656da14e0a40335c54a6b453c2d15184d4a2>    Bug is at line 288 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  288(c\_time\_10-cfg.dot) (time\_set\_timer)  …  -> 273(c\_time\_10-cfg.dot) (time\_set\_timer)  -> 259(py\_time\_test\_21-cfg.dot) (test\_set\_timer)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | Py\_INCREF | Foreign | Reference Counting | | PyInt\_Check | Foreign | Plain Integer Objects | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Cross-language function change |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the parsing arguments  Location: foreign function + if/branching statement |

| **Case 76** <https://github.com/pygame/pygame/commit/a36ccd65e50494e65fb241f55b31862f7578a737> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/2934> , which points to an earlier report at <https://github.com/pygame/pygame/issues/2912> 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/pygame/pygame/commit/a36ccd65e50494e65fb241f55b31862f7578a737>    Bug is at line 483 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  484(c\_base\_61-cfg.dot) (pg\_IntFromObjIndex)  …  -> 477(c\_base\_61-cfg.dot) (pg\_IntFromObjIndex)  -> 263(c\_rect\_6-cfg.dot) (pgRect\_FromObject)  …  -> 227(c\_rect\_6-cfg.dot) (pgRect\_FromObject)  -> 806(c\_rect\_27-cfg.dot) (pg\_rect\_clip)  -> 805(c\_rect\_27-cfg.dot) (pg\_rect\_clip)  -> 799(c\_rect\_27-cfg.dot) (pg\_rect\_clip)  -> 5157(py\_mask\_test\_204-cfg.dot) (test\_to\_surface\_\_area\_on\_mask\_with\_setsurface\_unsetsurface)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PySequence\_GetItem | Foreign | Sequence Protocol |   Data/control flow direction:   * Python->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: if/branching statement + foreign function |

| **Case 77** <https://github.com/pygame/pygame/commit/ef30264c6c85d4eeebcbbf3c9b10106782ed4a0b> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/2927> , which points to an earlier report at <https://github.com/pygame/pygame/issues/2862> 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/pygame/pygame/commit/ef30264c6c85d4eeebcbbf3c9b10106782ed4a0b>    Bug is at line 937 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  937(c\_draw\_21-cfg.dot) (rect)  …  -> 863(c\_draw\_21-cfg.dot) (rect)  -> 3146(py\_draw\_test\_254-cfg.dot) (test\_anti\_aliasing\_float\_coordinates)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | Py\_BuildValue | Foreign | Sequence Protocol | | PyErr\_Format | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 new function to handle  Location: if/branching statement + foreign function |

| **Case 78** <https://github.com/pygame/pygame/commit/fc3e6afca4c896a2b1c1e964eaf563e3cca21149> |
| --- |
| **Symptoms:**  There is an issue report on <https://github.com/pygame/pygame/pull/2880>. This report is directly associated with an earlier issue found here:<https://github.com/pygame/pygame/issues/2879> . Additionally, these reports are linked to a discussion on Stack Overflow <https://stackoverflow.com/questions/70209526/large-ellipse-draws-as-a-diamond> , 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/pygame/pygame/commit/fc3e6afca4c896a2b1c1e964eaf563e3cca21149>    Bug is at line 1894 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1894(c\_draw\_41-cfg.dot) (draw\_ellipse\_filled)  -> 1890(c\_draw\_41-cfg.dot) (draw\_ellipse\_filled)  -> 627(c\_draw\_18-cfg.dot) (ellipse)  ...  -> 582(c\_draw\_18-cfg.dot) (ellipse)  -> 447(py\_gfxdraw\_test\_20-cfg.dot) (test\_ellipse)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_Format | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: variable definition statement |

| **Case 79** <https://github.com/pygame/pygame/commit/bad58d0a7504ad1566504953a9c06ab2301c987e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/pull/1930> where it says    Thus, the symptom is error message (AssertionError) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/bad58d0a7504ad1566504953a9c06ab2301c987e>    Bug is at line 1249 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1249(c\_image\_11-cfg.dot) (image\_frombuffer)  …  -> 1216(c\_image\_11-cfg.dot) (image\_frombuffer)  -> 634(py\_image\_test\_29-cfg.dot) (test\_frombuffer\_RGBX)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 branch  Location: if/branching statement |

| **Case 80** <https://github.com/pygame/pygame/commit/ffd17758a238409a472bf13b00549bcc9647dac4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/1580> ,which points to an earlier report at <https://github.com/pygame/pygame/issues/1238> 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/pygame/pygame/commit/ffd17758a238409a472bf13b00549bcc9647dac4>    Bug is at line 1749 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1749(c\_event\_27-cfg.dot) (pg\_event\_post)  …  -> 1729(c\_event\_27-cfg.dot) (pg\_event\_post)  -> 329(py\_event\_test\_35-cfg.dot) (test\_get\_\_event\_sequence)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 new branch  Location: if/branching statement |

| **Case 81** <https://github.com/pygame/pygame/commit/1aeecd40314ec00c93d975b31a1f4518979d49a9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/565> 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/pygame/pygame/commit/1aeecd40314ec00c93d975b31a1f4518979d49a9>    Bug is at line 1418 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1418(c\_\_freetype\_99-cfg.dot) (\_ftfont\_getmetrics)  …  -> 1388(c\_\_freetype\_99-cfg.dot) (\_ftfont\_getmetrics)  -> 1257(py\_freetype\_test\_39-cfg.dot) (test\_undefined\_character\_code)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 variable  Location: if/branching statement + function call |

| **Case 82** <https://github.com/pygame/pygame/commit/20ff3f8e021d207b6053dd98eadf606c1eb8f451> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/numpy/numpy/pull/12683> 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/pygame/pygame/commit/20ff3f8e021d207b6053dd98eadf606c1eb8f451>    Bug is at line 1339 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1339(c\_event\_26-cfg.dot) (pg\_event\_get)  …  -> 1283(c\_event\_26-cfg.dot) (pg\_event\_get)  -> 195(py\_gen\_stubs\_7-cfg.dot) (get\_callables)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PySequence\_Check | Foreign | Sequence Protocol | | PySequence\_Size | Foreign | Sequence Protocol | | PyTuple\_Size | Foreign | Tuple Objects |   Data/control flow direction:   * Python->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 return statement  Location: if/branching statement + return statement + foreign function |

| **Case 83** <https://github.com/pygame/pygame/commit/40a77c6b9d2c13f017c63f00a85ef4d200b3be3e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/565> 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/pygame/pygame/commit/40a77c6b9d2c13f017c63f00a85ef4d200b3be3e>    Bug is at line 1419 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1419(c\_\_freetype\_99-cfg.dot) (\_ftfont\_getmetrics)  …  -> 1388(c\_\_freetype\_99-cfg.dot) (\_ftfont\_getmetrics)  -> 1257(py\_freetype\_test\_39-cfg.dot) (test\_undefined\_character\_code)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 variable  Location: conditional expression + function call |

| **Case 84** <https://github.com/pygame/pygame/commit/c3997fd84a3ed3608f59d71467363812b5870f90> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/1179> 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/pygame/pygame/commit/c3997fd84a3ed3608f59d71467363812b5870f90>    Bug is at line 213 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  213(c\_time\_10-cfg.dot) (time\_set\_timer)  …  -> 196(c\_time\_10-cfg.dot) (time\_set\_timer)  -> 259(py\_time\_test\_21-cfg.dot) (test\_set\_timer)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  index error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the index of array  Location: if/branching statement + function call |

| **Case 85** <https://github.com/pygame/pygame/commit/1513720270e118a957b8028121b12101c1c04f8b> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/179> 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/pygame/pygame/commit/1513720270e118a957b8028121b12101c1c04f8b>    Bug is at line 195 |
| **Bug manifestation:**  From the bug location, forward search until origin in Python:  path:  195(py\_sysfont\_7-cfg.dot) (initsysfonts\_macos)  -> 196(py\_sysfont\_7-cfg.dot) (initsysfonts\_macos)  -> 1033(c\_event\_25-cfg.dot) (event\_clear)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyTuple\_Size | Foreign | Tuple Objects | | PyTuple\_GET\_ITEM | Foreign | Tuple Objects | | PySequence\_Size | Foreign | Sequence Protocol | | PySequence\_Check | Foreign | Sequence Protocol |   Data/control flow direction:   * Python->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 variable  Location: function call |

| **Case 86** <https://github.com/pygame/pygame/commit/d33b702dcfdc3b13479e490961b7f51c7e773d33> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/313> 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/pygame/pygame/commit/d33b702dcfdc3b13479e490961b7f51c7e773d33>    Bug is at line 1617 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1614(c\_draw\_39-cfg.dot) (draw\_fillpoly)  …  -> 1556(c\_draw\_39-cfg.dot) (draw\_fillpoly)  -> 664(c\_draw\_20-cfg.dot) (polygon)  ...  -> 585(c\_draw\_20-cfg.dot) (polygon)  -> 415(py\_draw\_test\_44-cfg.dot) (test\_1\_pixel\_high\_or\_wide\_shapes)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyMem\_Free | Foreign | Memory Management | | PyMem\_New | Foreign | Memory Management | | PyErr\_NoMemory | Foreign | Exception Handling |   Data/control flow direction:   * Python->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 new branch  Location: if/branching statement |

| **Case 87** <https://github.com/pygame/pygame/commit/d5d5fc59d792bbeccc6535260d5d9fab61e84ffc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/313> where it says    Thus, the symptom is error message (assertion error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/d5d5fc59d792bbeccc6535260d5d9fab61e84ffc>    Bug is at line 181 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  180(py\_sysfont\_7-cfg.dot) (initsysfonts\_unix)  ...  -> 171(py\_sysfont\_7-cfg.dot) (initsysfonts\_unix)  -> 37(py\_sysfont\_1-cfg.dot) (toascii)  -> 41(py\_async\_sub\_5-cfg.dot) (decode)  ...  -> 40(py\_async\_sub\_5-cfg.dot) (decode)  -> 64(py\_rwobject\_test\_11-cfg.dot) (test\_string\_with\_null\_bytes)  -> 63(py\_rwobject\_test\_11-cfg.dot) (test\_string\_with\_null\_bytes)  -> 615(c\_rwobject\_18-cfg.dot) (rwobject\_encode\_string)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C->Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  String handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a string handling function  Location: loop statement |

| **Case 88** <https://github.com/pygame/pygame/commit/0e7f7b8327a18386843cf7baf4c24f07d826520e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/180> 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/pygame/pygame/commit/0e7f7b8327a18386843cf7baf4c24f07d826520e>    Bug is at line 2036 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2038(c\_surface\_122-cfg.dot) (surf\_get\_bounding\_rect)  ...  -> 1988(c\_surface\_122-cfg.dot) (surf\_get\_bounding\_rect)  -> 129(py\_surface\_test\_11-cfg.dot) (test\_get\_bounding\_rect)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PySurface\_Lock | NAX |  | | PySurface\_AsSurface | NAX |  |   Data/control flow direction:   * Python->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 switch statement to handle kinds of situations  Location: Function call |

| **Case 89** <https://github.com/pygame/pygame/commit/26a33627b8cfd28a31604345aaa8f9c0241b7e14> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/246> where it says    Thus, the symptom is error message (assertion error) |
| **Bug location:**  special case where SDL\_DisplayFormat was used was removed, with the known default intermediate surface format always used.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/26a33627b8cfd28a31604345aaa8f9c0241b7e14>      Bug is at line 3056 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  3052(c\_surface\_141-cfg.dot) (PySurface\_Blit)  ...  -> 2981(c\_surface\_141-cfg.dot) (PySurface\_Blit)  -> 1580(c\_surface\_102-cfg.dot) (surf\_blit)  ...  -> 1520(c\_surface\_102-cfg.dot) (surf\_blit)  -> 2248(py\_surface\_test\_213-cfg.dot) (test\_blanket\_alpha)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyImport\_ImportModule | Foreign | Importing Modules | | PySurface\_Prep | NAX |  |   Data/control flow direction:   * Python->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 switch statement to handle kinds of situations  Location: Function call |

| **Case 90** <https://github.com/pygame/pygame/commit/4c23a6ce02e4fa25ee62ba3d996106dcfd7d3cd4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/246> where it says    Thus, the symptom is error message (assertion error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/4c23a6ce02e4fa25ee62ba3d996106dcfd7d3cd4>    Bug is at line 1421 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1421(c\_\_freetype\_97-cfg.dot) (\_ftfont\_getrect)  ...  -> 1404(c\_\_freetype\_97-cfg.dot) (\_ftfont\_getrect)  -> 1106(py\_freetype\_test\_34-cfg.dot) (test\_freetype\_Font\_resolution)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyImport\_ImportModule | Foreign | Importing Modules | | PySurface\_Prep | NAX |  |   Data/control flow direction:   * Python->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 initial value  Location: assignment statement |

| **Case 91** <https://github.com/pygame/pygame/commit/95bf50120604bf9367f2df043d57f7fda81d49ba> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/243> 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/pygame/pygame/commit/95bf50120604bf9367f2df043d57f7fda81d49ba>    Bug is at line 456 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  374(c\_ft\_layout\_17-cfg.dot) (position\_glyphs)  ...  -> 277(c\_ft\_layout\_17-cfg.dot) (position\_glyphs)  -> 175(c\_ft\_layout\_14-cfg.dot) (\_PGFT\_LoadLayout)  ...  -> 139(c\_ft\_layout\_14-cfg.dot) (\_PGFT\_LoadLayout)  -> 256(c\_ft\_wrap\_17-cfg.dot) (\_PGFT\_GetTextRect)  -> 245(c\_ft\_wrap\_17-cfg.dot) (\_PGFT\_GetTextRect)  -> 1442(c\_\_freetype\_97-cfg.dot) (\_ftfont\_getrect)  …  -> 1404(c\_\_freetype\_97-cfg.dot) (\_ftfont\_getrect)  -> 1106(py\_freetype\_test\_34-cfg.dot) (test\_freetype\_Font\_resolution)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 if statement to handle values  Location: before function call |

| **Case 92** <https://github.com/pygame/pygame/commit/9f70cf7ffd26fa5268d33ba02ae6dfc924e63bc1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/165> where it says    Thus, the symptom is error message (resource warning) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/9f70cf7ffd26fa5268d33ba02ae6dfc924e63bc1>    Bug is at line 84,95 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  84(c\_display\_3-cfg.dot) (display\_resource)  ...  -> 46(c\_display\_3-cfg.dot) (display\_resource)  -> 541(c\_display\_20-cfg.dot) (set\_mode)  ...  -> 29(py\_image\_\_save\_gl\_surface\_test\_2-cfg.dot) (test\_image\_save\_works\_with\_opengl\_surfaces)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_INCREF | Foreign | Reference Counting | | Py\_DECREF | Foreign | Reference Counting | | PyErr\_Clear | Foreign | Exception Handling | | PyObject\_CallFunction | Foreign | Call Protocol | | PyObject\_GetAttrString | Foreign | Object Protocol | | PyImport\_ImportModule | Foreign | Importing Modules |   Data/control flow direction:   * Python->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 a function to handle the reference count  Location: after foreign function + if/branching statement |

| **Case 93** <https://github.com/pygame/pygame/commit/9f70cf7ffd26fa5268d33ba02ae6dfc924e63bc1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/8> 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/pygame/pygame/commit/9f70cf7ffd26fa5268d33ba02ae6dfc924e63bc1>    Bug is at line 231 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  228(py\_cursors\_1-cfg.dot) (compile)  ...  -> 223(py\_cursors\_1-cfg.dot) (compile)  -> 307(py\_gen\_stubs\_19-cfg.dot) (already\_tested\_in\_package)  -> 169(py\_gen\_stubs\_4-cfg.dot) (get\_package\_modules)  -> 165(py\_gen\_stubs\_3-cfg.dot) (module\_in\_package)  -> 164(py\_gen\_stubs\_3-cfg.dot) (module\_in\_package)  -> 169(py\_gen\_stubs\_4-cfg.dot) (get\_package\_modules)  -> 168(py\_gen\_stubs\_4-cfg.dot) (get\_package\_modules)  -> 186(py\_gen\_stubs\_6-cfg.dot) (is\_public)  -> 184(py\_gen\_stubs\_6-cfg.dot) (is\_public)  -> 183(py\_gen\_stubs\_6-cfg.dot) (is\_public)  -> 168(py\_gen\_stubs\_4-cfg.dot) (get\_package\_modules)  -> 167(py\_gen\_stubs\_4-cfg.dot) (get\_package\_modules)  -> 307(py\_gen\_stubs\_19-cfg.dot) (already\_tested\_in\_package)  -> 305(py\_gen\_stubs\_19-cfg.dot) (already\_tested\_in\_package)  -> 304(py\_gen\_stubs\_19-cfg.dot) (already\_tested\_in\_package)  -> 328(py\_gen\_stubs\_20-cfg.dot) (get\_stubs)  -> 327(py\_gen\_stubs\_20-cfg.dot) (get\_stubs)  -> 264(py\_gen\_stubs\_15-cfg.dot) (package\_stubs)  -> 261(py\_gen\_stubs\_15-cfg.dot) (package\_stubs)  -> 262(py\_gen\_stubs\_15-cfg.dot) (package\_stubs)  -> 256(py\_gen\_stubs\_14-cfg.dot) (module\_stubs)  -> 254(py\_gen\_stubs\_14-cfg.dot) (module\_stubs)  -> 235(py\_gen\_stubs\_13-cfg.dot) (make\_stubs)  -> 232(py\_gen\_stubs\_12-cfg.dot) (test\_stub)  -> 219(py\_gen\_stubs\_12-cfg.dot) (test\_stub)  -> 1466(c\_\_freetype\_73-cfg.dot) (\_ftface\_render)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * C ->Python |
| **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: move a statement to correct place  Location: if/branching statement |

| **Case 94** <https://github.com/pygame/pygame/commit/c7ec15285c23ce54f466c9a3482b4d087eb8b0ff> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/8> 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/pygame/pygame/commit/c7ec15285c23ce54f466c9a3482b4d087eb8b0ff>    Bug is at line 1439 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1439(c\_surface\_90-cfg.dot) (surf\_fill)  ...  -> 1392(c\_surface\_90-cfg.dot) (surf\_fill)  -> 172(py\_transform\_test\_6-cfg.dot) (test\_threshold\_\_honors\_third\_surface)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyLong\_Check | Foreign | Integer Objects | | PyInt\_Check | Foreign | Plain Integer Objects | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python ->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 95** <https://github.com/giampaolo/psutil/commit/b88f4d839dc4d665526e91b098d018a3d796a0ac> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/2138> where it says    Thus, the symptom is error message (import error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/b88f4d839dc4d665526e91b098d018a3d796a0ac>    Bug is at line 444 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  444(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 441(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 440(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 438(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 437(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 436(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 435(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 432(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 430(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 429(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 426(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 424(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 417(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 414(c\_\_psutil\_linux\_4-cfg.dot) (psutil\_net\_if\_duplex\_speed)  -> 1064(py\_\_pslinux\_43-cfg.dot) (net\_if\_stats)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromOSErrnoWithSyscall | NAX | Exception Handling |   Data/control flow direction:   * Python ->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Compatibility Issue |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: move function from header file to c file  Location: assignment statement |

| **Case 96** <https://github.com/giampaolo/psutil/commit/19461905377ce3060b60df1949c082086492dab9#> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/2138> 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/giampaolo/psutil/commit/19461905377ce3060b60df1949c082086492dab9#>    Bug is at line 250 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  250(c\_proc\_6-cfg.dot) (psutil\_proc\_threads)  …  -> 203(c\_proc\_6-cfg.dot) (psutil\_proc\_threads)  -> 538(py\_\_psosx\_47-cfg.dot) (threads)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_SetFromErrno | Foreign | Exception Handling | | PyList\_New | Foreign | List Objects |   Data/control flow direction:   * Python ->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: loop statement |

| **Case 97** <https://github.com/giampaolo/psutil/commit/2963b5f264b451a67dd26e19d38ab890a5bafa41> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/2138> where it says    Thus, the symptom is error message (EBADF error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/2963b5f264b451a67dd26e19d38ab890a5bafa41>    Bug is at line 357 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  357(c\_[\_psutil\_posix](https://github.com/giampaolo/psutil/commit/2963b5f264b451a67dd26e19d38ab890a5bafa41#diff-9028265a1a12dbebb13fa6d060ab6c3108bd3df180ee719374ad871ce130be0b)\_7-cfg.dot) (psutil\_net\_if\_mtu)  …  -> 354(c\_[\_psutil\_posix](https://github.com/giampaolo/psutil/commit/2963b5f264b451a67dd26e19d38ab890a5bafa41#diff-9028265a1a12dbebb13fa6d060ab6c3108bd3df180ee719374ad871ce130be0b)\_7-cfg.dot) (psutil\_net\_if\_mtu)  -> 538(py\_\_psosx\_47-cfg.dot) (net\_if\_stats)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python ->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 initial value  Location: assignment statement |

| **Case 98** <https://github.com/giampaolo/psutil/commit/72c84cb4edb5c0968a83c1f45ad5cc51235e0af3> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1595> 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/giampaolo/psutil/commit/72c84cb4edb5c0968a83c1f45ad5cc51235e0af3>    Bug is at line 280 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  269(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 255(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 254(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 251(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 249(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 243(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 892(py\_\_pswindows\_88-cfg.dot) (kill)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromOSErrnoWithSyscall | NAX | Exception handling |   Data/control flow direction:   * Python ->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 exception handling  Location: if/branching statement |

| **Case 99** <https://github.com/giampaolo/psutil/commit/b2414b83d3d728ec34ea0e35bfb21517ee231401> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1595> 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/giampaolo/psutil/commit/b2414b83d3d728ec34ea0e35bfb21517ee231401>    Bug is at line 283 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  -> 254(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 253(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 250(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 248(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 243(c\_\_psutil\_windows\_6-cfg.dot) (psutil\_proc\_kill)  -> 892(py\_\_pswindows\_88-cfg.dot) (kill)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromOSErrnoWithSyscall | NAX | Exception handling |   Data/control flow direction:   * Python ->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 100** <https://github.com/giampaolo/psutil/commit/b4c2124ff1f8f4f1f7f50ad0af6945b0d7e945df> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1823> 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/giampaolo/psutil/commit/b4c2124ff1f8f4f1f7f50ad0af6945b0d7e945df>    Bug is at line 283 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  250(c\_process\_handles\_4-cfg.dot) (psutil\_get\_open\_files)  ...  -> 209(c\_process\_handles\_4-cfg.dot) (psutil\_get\_open\_files)  -> 806(c\_\_psutil\_windows\_18-cfg.dot) (psutil\_proc\_open\_files)  ...  -> 792(c\_\_psutil\_windows\_18-cfg.dot) (psutil\_proc\_open\_files)  -> 853(py\_\_psbsd\_54-cfg.dot) (open\_files)  -> 851(py\_\_psbsd\_54-cfg.dot) (open\_files)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyUnicode\_FromWideChar | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * Python ->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 101** <https://github.com/giampaolo/psutil/commit/d180e25b871c314614aa96eccadb0bcc0a5df676> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1439> 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/giampaolo/psutil/commit/d180e25b871c314614aa96eccadb0bcc0a5df676>        Bug is at line 812 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  812(py\_\_psbsd\_43-cfg.dot) (connections)  -> 624(py\_\_psbsd\_29-cfg.dot) (name)  -> 371(c\_\_psutil\_bsd\_6-cfg.dot) (psutil\_proc\_name)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyUnicode\_DecodeFSDefault | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * C ->Python |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  ? |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: conditional expression |

| **Case 102** <https://github.com/giampaolo/psutil/commit/7381d4d15dd7fe108e1bf350c3f76035922c6ed4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1439> where it says    Thus, the symptom is error message (windows error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/7381d4d15dd7fe108e1bf350c3f76035922c6ed4>    Bug is at line 773 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  773(c\_\_psutil\_windows\_22-cfg.dot) (psutil\_proc\_exe)  -> 771(c\_\_psutil\_windows\_22-cfg.dot) (psutil\_proc\_exe)  -> 770(c\_\_psutiltraverws\_22-cfg.dot) (psutil\_proc\_exe)  -> 768(c\_\_psutil\_windows\_22-cfg.dot) (psutil\_proc\_exe)  -> 762(c\_\_psutil\_windows\_22-cfg.dot) (psutil\_proc\_exe)  -> 705(py\_\_pswindows\_70-cfg.dot) (exe)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromWindowsErr | Foreign | Exception Handling |   Data/control flow direction:   * Python->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 + foreign function  Location: conditional expression |

| **Case 103** <https://github.com/giampaolo/psutil/commit/842a50538fb518291079ab5f245cca4c63813cb2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1439> 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/giampaolo/psutil/commit/842a50538fb518291079ab5f245cca4c63813cb2>    Bug is at line 346 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  346(c\_\_psutil\_bsd\_3-cfg.dot) (psutil\_proc\_oneshot\_info)  …  -> 193(c\_\_psutil\_bsd\_3-cfg.dot) (psutil\_proc\_oneshot\_info)  -> 611(py\_\_psbsd\_26-cfg.dot) (oneshot)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyUnicode\_DecodeFSDefault | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * Python->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: Boundary check against value range/limit + foreign function  Location: conditional expression |

| **Case 104** <https://github.com/giampaolo/psutil/commit/9c75262c01266a3e2e31cd09d5d9c710d0270e18> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1127> where it says    Thus, the symptom is error massage (EPERM error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/9c75262c01266a3e2e31cd09d5d9c710d0270e18>    Bug is at line 1170 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1132(c\_\_psutil\_osx\_23-cfg.dot) (psutil\_proc\_open\_files)  -> 1131(c\_\_psutil\_osx\_23-cfg.dot) (psutil\_proc\_open\_files)  -> 1129(c\_\_psutil\_osx\_23-cfg.dot) (psutil\_proc\_open\_files)  -> 1126(c\_\_psutil\_osx\_23-cfg.dot) (psutil\_proc\_open\_files)  -> 1124(c\_\_psutil\_osx\_23-cfg.dot) (psutil\_proc\_open\_files)  -> 358(c\_process\_info\_6-cfg.dot) (psutil\_proc\_pidinfo)  -> 357(c\_process\_info\_6-cfg.dot) (psutil\_proc\_pidinfo)  -> 125(c\_\_psutil\_posix\_2-cfg.dot) (psutil\_raise\_for\_pid)  -> 122(c\_\_psutil\_posix\_2-cfg.dot) (psutil\_raise\_for\_pid)  ...  -> 122(c\_\_psutil\_posix\_2-cfg.dot) (psutil\_raise\_for\_pid)  -> 121(c\_\_psutil\_posix\_2-cfg.dot) (psutil\_raise\_for\_pid)  -> 277(c\_\_psutil\_windows\_11-cfg.dot) (psutil\_pid\_exists)  -> 276(c\_\_psutil\_windows\_11-cfg.dot) (psutil\_pid\_exists)  -> 271(c\_\_psutil\_windows\_11-cfg.dot) (psutil\_pid\_exists)  -> 240(c\_process\_info\_4-cfg.dot) (psutil\_get\_environ)  -> 237(c\_process\_info\_4-cfg.dot) (psutil\_get\_environ)  -> 54(py\_test\_bsd\_1-cfg.dot) (sysctl)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyUnicode\_DecodeFSDefault | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * Python->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: After foreign function to make the variable to NULL  Location: foreign function |

| **Case 105** <https://github.com/giampaolo/psutil/commit/ab90e4e6ac73c249cf7aea7e92aec2b6a07ef041> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1127> 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/giampaolo/psutil/commit/ab90e4e6ac73c249cf7aea7e92aec2b6a07ef041>    Bug is at line 2405 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  2405(c\_\_psutil\_windows\_50-cfg.dot) (psutil\_disk\_io\_counters)  …  -> 2361(c\_\_psutil\_windows\_50-cfg.dot) (psutil\_disk\_io\_counters)  -> 1155(py\_test\_linux\_142-cfg.dot) (test\_sector\_size\_mock)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyDict\_SetItemString | Foreign | Dictionary Objects | | Py\_XDECREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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 and if statement to handle the special case  Location: if/branching statement |

| **Case 106** <https://github.com/giampaolo/psutil/commit/a759a044123c88a856aedb1e1994e8944981ecdf> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1033> 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/giampaolo/psutil/commit/a759a044123c88a856aedb1e1994e8944981ecdf>    Bug is at line 1376 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1376(c\_\_psutil\_osx\_24-cfg.dot) (psutil\_proc\_connections)  …  -> 1196(c\_\_psutil\_osx\_24-cfg.dot) (psutil\_proc\_connections)  -> 457(py\_\_psosx\_38-cfg.dot) (connections)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyUnicode\_DecodeFSDefault | Foreign | Unicode Objects and Codecs | | PyList\_Append | Foreign | List Objects | | Py\_DECREF | Foreign | Reference Counting | | PyErr\_SetFromErrno | Foreign | Exception Handling | | PySequence\_Contains | Foreign | Sequence Protocol | | PyLong\_FromLong | Foreign | Integer Objects |   Data/control flow direction:   * Python->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 foreign function to manage reference counting  Location: After foreign function |

| **Case 107** <https://github.com/giampaolo/psutil/commit/fd2205bad09db0efab8958c359897fc1772dc187> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/1069> 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/giampaolo/psutil/commit/fd2205bad09db0efab8958c359897fc1772dc187>    Bug is at line 200 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  200(c\_\_psutil\_bsd\_3-cfg.dot) (psutil\_proc\_oneshot\_info)  -> 192(c\_\_psutil\_bsd\_3-cfg.dot) (psutil\_proc\_oneshot\_info)  -> 546(py\_\_psbsd\_24-cfg.dot) (oneshot)  -> 544(py\_\_psbsd\_24-cfg.dot) (oneshot)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyUnicode\_DecodeFSDefault | Foreign | Unicode Objects and Codecs |   Data/control flow direction:   * Python->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: variable initialization |

| **Case 108** <https://github.com/giampaolo/psutil/commit/69a273209a41f3c51bdc22d12c9f833ccc68dd55> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/961> where it says    Thus, the symptom is windows services.msc fails |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/69a273209a41f3c51bdc22d12c9f833ccc68dd55>    Bug is at line 379 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  379(c\_services\_7-cfg.dot) (psutil\_winservice\_query\_descr)  …  -> 355(c\_services\_7-cfg.dot) (psutil\_winservice\_query\_descr)  -> 515(py\_\_pswindows\_41-cfg.dot) (description)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: handle the exception  Location: after function call |

| **Case 109** <https://github.com/giampaolo/psutil/commit/ca8b7f49cd2a7c1dd83e514b15e28ab709903227> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/933> 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/giampaolo/psutil/commit/ca8b7f49cd2a7c1dd83e514b15e28ab709903227>    Bug is at line 385 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  385(c\_services\_7-cfg.dot) (psutil\_winservice\_query\_descr)  …  -> 355(c\_services\_7-cfg.dot) (psutil\_winservice\_query\_descr)  -> 487(py\_\_pswindows\_40-cfg.dot) (description)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromWindowsErr | Foreign | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  service handler management |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: call a function to close the handler  Location: after using the service handler |

| **Case 110** <https://github.com/giampaolo/psutil/commit/852cb32371d1f285b75b8ec70a60210dbfa169c6> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/984> 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/giampaolo/psutil/commit/852cb32371d1f285b75b8ec70a60210dbfa169c6>    Bug is at line 305 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  305(c\_process\_handles\_5-cfg.dot) (psutil\_NtQueryObject)  ...  -> 274(c\_process\_handles\_5-cfg.dot) (psutil\_NtQueryObject)  -> 186(c\_process\_handles\_4-cfg.dot) (psutil\_get\_open\_files\_ntqueryobject)  ...  -> 67(c\_process\_handles\_4-cfg.dot) (psutil\_get\_open\_files\_ntqueryobject)  -> 40(c\_process\_handles\_2-cfg.dot) (psutil\_get\_open\_files)  ...  -> 30(c\_process\_handles\_2-cfg.dot) (psutil\_get\_open\_files)  -> 1275(c\_\_psutil\_windows\_34-cfg.dot) (psutil\_proc\_open\_files)  ...  -> 1262(c\_\_psutil\_windows\_34-cfg.dot) (psutil\_proc\_open\_files)  -> 852(py\_\_pswindows\_89-cfg.dot) (open\_files)  -> 847(py\_\_pswindows\_89-cfg.dot) (open\_files)  -> 845(py\_\_pswindows\_89-cfg.dot) (open\_files)  -> 844(py\_\_pswindows\_89-cfg.dot) (open\_files)  -> 1250(py\_test\_linux\_173-cfg.dot) (test\_open\_files\_file\_gone)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyErr\_NoMemory | Foreign | Exception Handling | | PyErr\_SetFromWindowsErr | Foreign | Exception Handling | | PyList\_New | Foreign | List Objects |   Data/control flow direction:   * Python->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: delete the misused function call  Location: if/branching statement + function call |

| **Case 111** <https://github.com/giampaolo/psutil/commit/5f02dd0c04576633bd0740c756bef3a64efe3ebc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/740> where it says    Thus, the symptom is warning message (incompatible types) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/5f02dd0c04576633bd0740c756bef3a64efe3ebc>    Bug is at line 3065 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  3065(c\_\_psutil\_windows\_56-cfg.dot) (psutil\_net\_if\_stats)  …  -> 2971(c\_\_psutil\_windows\_56-cfg.dot) (psutil\_net\_if\_stats)  -> 316(py\_\_windows\_27-cfg.dot) (test\_net\_if\_stats)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyDict\_SetItemString | Foreign | Dictionary Objects | | Py\_BuildValue | Foreign | Parsing arguments and building values | | Py\_INCREF | Foreign | Reference Counting | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling | | PyDict\_New | Foreign | Dictionary Objects |   Data/control flow direction:   * Python->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 misused function call  Location: conditional expression + foreign function call |

| **Case 112** <https://github.com/giampaolo/psutil/commit/984f060eefb328310060e0ec480c5c887b0f9704> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/774> where it says    Thus, the symptom is no descripetion |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/984f060eefb328310060e0ec480c5c887b0f9704>    Bug is at line 815 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  -> 807(c\_\_psutil\_sunos\_14-cfg.dot) (psutil\_net\_io\_counters)  …  -> 752(c\_\_psutil\_sunos\_14-cfg.dot) (psutil\_net\_io\_counters)  -> 261(py\_test\_posix\_36-cfg.dot) (test\_nic\_names)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyErr\_SetFromErrno | Foreign | Exception Handling | | PyErr\_NoMemory | Foreign | Exception Handling |   Data/control flow direction:   * Python->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 the error argument  Location: foreign function call |

| **Case 113** <https://github.com/giampaolo/psutil/commit/08c6017fbb2820e7cefed8b0f41f59877969f7f3> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/707> 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/giampaolo/psutil/commit/08c6017fbb2820e7cefed8b0f41f59877969f7f3>    Bug is at line 835 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  835(c\_\_psutil\_sunos\_15-cfg.dot) (psutil\_net\_connections)  …  -> 804(c\_\_psutil\_sunos\_15-cfg.dot) (psutil\_net\_connections)  -> 216(py\_\_pssunos\_16-cfg.dot) (net\_connections)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 error argument  Location: foreign function call |

| **Case 114** <https://github.com/giampaolo/psutil/commit/241c6baeaf2c59c609f47586a0404cd6c619d205> |
| --- |
| **Symptoms:**  No description |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/241c6baeaf2c59c609f47586a0404cd6c619d205>    Bug is at line 339,340 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  340(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 339(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 336(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 332(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 427(py\_\_psbsd\_42-cfg.dot) (cpu\_affinity\_get)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromErrno | Foreign | Exception Handling | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyList\_New | Foreign | List Objects | | PyList\_Append | Foreign | List Objects | | Py\_DECREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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: give the NULL value to the pointer variable when doing the definition  Location: assignment statement |

| **Case 115** <https://github.com/giampaolo/psutil/commit/998ceccf9371bc13ec095c5b2941968971dfe33a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/664> where it says    Thus, the symptom is error message (undeclared identifier) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/998ceccf9371bc13ec095c5b2941968971dfe33a>    Bug is at line 521 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  521(c\_\_psutil\_linux\_6-cfg.dot) (psutil\_net\_if\_stats)  -> 480(c\_\_psutil\_linux\_6-cfg.dot) (psutil\_net\_if\_stats)  …  -> 217(py\_\_pswindows\_16-cfg.dot) (net\_if\_stats)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_INCREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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 116** <https://github.com/giampaolo/psutil/commit/998ceccf9371bc13ec095c5b2941968971dfe33a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/527> 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/giampaolo/psutil/commit/998ceccf9371bc13ec095c5b2941968971dfe33a>    Bug is at line 268 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  268(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  …  -> 255(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 1061(py\_\_pslinux\_59-cfg.dot) (cpu\_affinity\_get)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 extra code including if, loop to handle the special case  Location: return statement + foreign function |

| **Case 117** <https://github.com/giampaolo/psutil/commit/7f9cb6ec8d975b80957b77c2c5fbeca07236e170> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/561> 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/giampaolo/psutil/commit/7f9cb6ec8d975b80957b77c2c5fbeca07236e170>    Bug is at line 452 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  452(py\_\_pslinux\_19-cfg.dot) (process\_inet)  ...  -> 438(py\_\_pslinux\_19-cfg.dot) (process\_inet)  -> 509(py\_\_pslinux\_21-cfg.dot) (retrieve)  ...  -> 495(py\_\_pslinux\_21-cfg.dot) (retrieve)  -> 1066(py\_\_pslinux\_66-cfg.dot) (connections)  -> 1065(py\_\_pslinux\_66-cfg.dot) (connections)  -> 1840(py\_test\_psutil\_161-cfg.dot) (test\_connections\_all)  -> 1839(py\_test\_psutil\_161-cfg.dot) (test\_connections\_all)  -> 807(py\_\_\_init\_\_\_61-cfg.dot) (children)  ...  -> 743(py\_\_\_init\_\_\_61-cfg.dot) (children)  -> 2999(c\_\_psutil\_windows\_57-cfg.dot) (psutil\_ppid\_map)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_DECREF | Foreign | Reference Counting | | Py\_XDECREF | Foreign | Reference Counting | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyErr\_SetFromWindowsErr | Foreign | Exception Handling |   Data/control flow direction:   * C->Python |
| **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 variable name  Location: conditional expression |

| **Case 118** <https://github.com/giampaolo/psutil/commit/84171a34b489fb77631233b14060222c2975ab12> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/527> 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/giampaolo/psutil/commit/84171a34b489fb77631233b14060222c2975ab12>    Bug is at line 83 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  83(py\_process\_detail\_3-cfg.dot) (run)  -> 82(py\_process\_detail\_3-cfg.dot) (run)  -> 77(py\_process\_detail\_3-cfg.dot) (run)  -> 263(py\_\_pssunos\_17-cfg.dot) (name)  -> 120(c\_\_psutil\_sunos\_3-cfg.dot) (psutil\_proc\_name\_and\_args)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C->Python |
| **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: function call |

| **Case 119** <https://github.com/giampaolo/psutil/commit/fc024a3cc2f95b46605c8f7c48c64ea6b3f25af5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/pull/537> , which points to an earlier report at <https://github.com/giampaolo/psutil/issues/522> where it says    Thus, the symptom is error message (OS error/Invalid argument) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/giampaolo/psutil/commit/fc024a3cc2f95b46605c8f7c48c64ea6b3f25af5>    Bug is at line 327 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  327(c\_\_psutil\_linux\_3-cfg.dot) (psutil\_proc\_cpu\_affinity\_get)  -> 1067(py\_\_pslinux\_59-cfg.dot) (cpu\_affinity\_get)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyErr\_SetFromErrno | Foreign | Exception Handling | | PyList\_Append | Foreign | List Objects | | PyList\_New | Foreign | List Objects |   Data/control flow direction:   * Python->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 same name function  Location: function call |

| **Case 120** <https://github.com/pygame/pygame/commit/0fee6ea576e8ea109369d99ff19c02170d398eb5> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/120> where it says    Thus, the symptom is error message (type error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/0fee6ea576e8ea109369d99ff19c02170d398eb5>    Bug is at line 133 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  133(c\_\_numericsurfarray\_2-cfg.dot) (pixels2d)  …  -> 101(c\_\_numericsurfarray\_2-cfg.dot) (pixels2d)  -> 764(py\_surfarray\_test\_35-cfg.dot) (test\_surf\_lock)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | numpy\_array | NAX | Array API |   Data/control flow direction:   * Python->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: add a type of error  Location: except statement |

| **Case 121** <https://github.com/pygame/pygame/commit/24682c6c2033f45eb1328eac81d58d82d7127044> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/146> where it says    Thus, the symptom is error message (attribute error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/24682c6c2033f45eb1328eac81d58d82d7127044>    Bug is at line 58 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  58(py\_pkgdata\_3-cfg.dot) (getResource)  -> 59(py\_pkgdata\_3-cfg.dot) (getResource)  -> 66(py\_pkgdata\_3-cfg.dot) (getResource)  -> 45(py\_image\_test\_3-cfg.dot) (testLoadIcon)  -> 46(py\_image\_test\_3-cfg.dot) (testLoadIcon)  -> 48(py\_image\_test\_3-cfg.dot) (testLoadIcon)  -> 51(c\_image\_4-cfg.dot) (image\_load\_basic)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_DECREF | Foreign | Reference Counting |   Data/control flow direction:   * Python->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: assignment statement + function call |

| **Case 122** <https://github.com/pygame/pygame/commit/83c1c1c048a17cb3fac7309c02b3e54af2870ad4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/commit/83c1c1c048a17cb3fac7309c02b3e54af2870ad4> where it says    Thus, the symptom is error message (overflow error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/83c1c1c048a17cb3fac7309c02b3e54af2870ad4>    Bug is at line 1610 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1610(c\_color\_95-cfg.dot) (\_color\_set\_length)  -> 1605(c\_color\_95-cfg.dot) (\_color\_set\_length)  -> 795(py\_color\_test\_73-cfg.dot) (test\_arraystruct)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: handle the other data type by if statement  Location: conditional expression + foreign function call |

| **Case 123** <https://github.com/pygame/pygame/commit/fe2dcee303773d2496685fd3c1b91c9e0ea93613> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygame/pygame/issues/144> where it says    Thus, the symptom is error message (system error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygame/pygame/commit/fe2dcee303773d2496685fd3c1b91c9e0ea93613>    Bug is at line 234,235 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  234(c\_ft\_render\_6-cfg.dot) (\_PGFT\_GetRenderMetrics)  ...  -> 198(c\_ft\_render\_6-cfg.dot) (\_PGFT\_GetRenderMetrics)  -> 682(c\_ft\_render\_8-cfg.dot) (\_PGFT\_Render\_Array)  ...  -> 656(c\_ft\_render\_8-cfg.dot) (\_PGFT\_Render\_Array)  -> 98(c\_pixelcopy\_2-cfg.dot) (\_validate\_view\_format)  ...  -> 43(c\_pixelcopy\_2-cfg.dot) (\_validate\_view\_format)  -> 781(c\_pixelcopy\_9-cfg.dot) (surface\_to\_array)  ...  -> 752(c\_pixelcopy\_9-cfg.dot) (surface\_to\_array)  -> 622(py\_pixelcopy\_test\_63-cfg.dot) (test\_surface\_to\_array\_newbuf)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: conditional expression + foreign function call |

| **Case 124** <https://github.com/giampaolo/psutil/commit/46460516c8579149d9959454040f453c1a216472> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/63> 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/giampaolo/psutil/commit/46460516c8579149d9959454040f453c1a216472>    Bug is at line 701 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  701(c\_\_psutil\_mswindows\_21-cfg.dot) (get\_proc\_username)  …  -> 721(c\_\_psutil\_mswindows\_21-cfg.dot) (get\_proc\_username)  -> 98(py\_\_psmswindows\_13-cfg.dot) (get\_process\_username)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 + function call |

| **Case 125** <https://github.com/giampaolo/psutil/commit/46460516c8579149d9959454040f453c1a216472> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/63> 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/giampaolo/psutil/commit/46460516c8579149d9959454040f453c1a216472>    Bug is at line 701 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  701(c\_\_psutil\_mswindows\_21-cfg.dot) (get\_proc\_username)  …  -> 721(c\_\_psutil\_mswindows\_21-cfg.dot) (get\_proc\_username)  -> 98(py\_\_psmswindows\_13-cfg.dot) (get\_process\_username)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 + function call |

| **Case 126** <https://github.com/giampaolo/psutil/commit/73a84d21d61bc66d90b8a2ec86c44fa92171bc0d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/232> 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/giampaolo/psutil/commit/73a84d21d61bc66d90b8a2ec86c44fa92171bc0d>    Bug is at line 144 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  144(c\_\_psutil\_linux\_2-cfg.dot) (get\_physmem)  …  -> 137(c\_\_psutil\_linux\_2-cfg.dot) (get\_physmem)  -> 126(py\_\_pslinux\_4-cfg.dot) (phymem\_buffers)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyErr\_SetFromErrno | Foreign | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument variable in the foreign function  Location: return statement + foreign function call |

| **Case 127** <https://github.com/giampaolo/psutil/commit/85209aa163634d149d8ea9806b7e095e4e29bb0d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/102> 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/giampaolo/psutil/commit/85209aa163634d149d8ea9806b7e095e4e29bb0d>    Bug is at line 312, 320 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  312(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 310(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 309(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 308(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 307(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 303(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 297(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 295(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 290(c\_\_psutil\_mswindows\_7-cfg.dot) (get\_process\_cpu\_times)  -> 92(py\_\_psosx\_14-cfg.dot) (get\_cpu\_times)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_BuildValue | Foreign | Parsing arguments and building values | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_SetFromWindowsErr | Foreign | Exception Handling | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * Python->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: return statement |

| **Case 128** <https://github.com/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/200> 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/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658>    Bug is at line 31 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  31(c\_allisinstance\_3-cfg.dot) (PyIU\_AllIsinstance)  …  -> 5(c\_allisinstance\_3-cfg.dot) (PyIU\_AllIsinstance)  -> 76(py\_test\_allisinstance\_34-cfg.dot) (test\_allisinstance\_failure5)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_TYPE | Foreign | Common Object Structures | | PyObject\_GetIter | Foreign | Object Protocol |   Data/control flow direction:   * Python->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: Move initialization into loop body  Location: assignment statement + foreign function call + loop |

| **Case 129** <https://github.com/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/200> 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/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658>    Bug is at line 31 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  31(c\_anyisinstance\_4-cfg.dot) (PyIU\_AllIsinstance)  …  -> 5(c\_anyisinstance\_4-cfg.dot) (PyIU\_AllIsinstance)  -> 81(py\_test\_anyisinstance\_35-cfg.dot) (test\_anyisinstance\_failure5)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_TYPE | Foreign | Common Object Structures | | PyObject\_GetIter | Foreign | Object Protocol |   Data/control flow direction:   * Python->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: Move initialization into loop body  Location: assignment statement + foreign function call + loop |

| **Case 130** <https://github.com/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/200> 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/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658>    Bug is at line 72 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  72(c\_groupedby\_9-cfg.dot) (PyIU\_Groupby)  …  -> 5(c\_groupedby\_9-cfg.dot) (PyIU\_Groupby)  -> 159(py\_test\_groupedby\_39-cfg.dot) (test\_groupedby\_failure10)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_TYPE | Foreign | Common Object Structures | | PyObject\_GetIter | Foreign | Object Protocol |   Data/control flow direction:   * Python->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: Move initialization into loop body  Location: assignment statement + foreign function call + loop |

| **Case 131** <https://github.com/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/200> 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/MSeifert04/iteration_utilities/commit/119cfcffe1885fca178e15710f084b25d387a658>    Bug is at line 154 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  154(c\_grouper\_10-cfg.dot) (grouper\_next)  …  -> 121(c\_grouper\_10-cfg.dot) (grouper\_next)  -> 216(py\_test\_grouper\_39-cfg.dot) (test\_grouper\_failure7)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_TYPE | Foreign | Common Object Structures | | PyObject\_GetIter | Foreign | Object Protocol | | Py\_REFCNT | Foreign | Reference Counting |   Data/control flow direction:   * Python->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: Move initialization into loop body  Location: assignment statement + foreign function call + loop |

| **Case 132** <https://github.com/MSeifert04/iteration_utilities/commit/dc5e033ad648e5373ae1f34d4409380bbd4ab8a1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/198> 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/MSeifert04/iteration_utilities/commit/dc5e033ad648e5373ae1f34d4409380bbd4ab8a1>    Bug is at line 202 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  202(c\_accumulate\_5-cfg.dot) (accumulate\_lengthhint)  …  -> 199(c\_accumulate\_5-cfg.dot) (accumulate\_lengthhint)  -> 173(py\_test\_accumulate\_41-cfg.dot) (test\_accumulate\_lengthhint1)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyLong\_FromSsize\_t | Foreign | Integer Objects | | PyObject\_LengthHint | Foreign | bject Protocol |   Data/control flow direction:   * Python->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 + return Null value  Location: if/branching statement + function call |

| **Case 133** <https://github.com/MSeifert04/iteration_utilities/commit/dc5e033ad648e5373ae1f34d4409380bbd4ab8a1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/198> 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/MSeifert04/iteration_utilities/commit/dc5e033ad648e5373ae1f34d4409380bbd4ab8a1>    Bug is at line 223 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  202(c\_replicate\_11-cfg.dot) (replicate\_lengthhint)  …  -> 199(c\_replicate\_11-cfg.dot) (replicate\_lengthhint)  -> 162(py\_test\_replicate\_43-cfg.dot) (test\_replicate\_failure\_lengthhint1)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyLong\_FromSsize\_t | Foreign | Integer Objects | | PyObject\_LengthHint | Foreign | bject Protocol | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Integer overflow |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: type conversion  Location: assignment statement + return |

| **Case 134** <https://github.com/MSeifert04/iteration_utilities/commit/1361a3a3e1c0d75dd4ee2ad2609d2f795bcdfc6f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/giampaolo/psutil/issues/102> 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/MSeifert04/iteration_utilities/commit/1361a3a3e1c0d75dd4ee2ad2609d2f795bcdfc6f>    Bug is at line 170 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  170(c\_deepflatten\_12-cfg.dot) (deepflatten\_next)  …  -> 117(c\_deepflatten\_12-cfg.dot) (deepflatten\_next)  -> 216(py\_test\_deepflatten\_43-cfg.dot) (test\_deepflatten\_failure9)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_IsInstance | Foreign | Object Protocol | | PyList\_GET\_ITEM | Foreign | List Objects | | PyList\_SET\_ITEM | Foreign | List Objects | | PyErr\_SetString | Foreign | Exception Handling | | PyErr\_ExceptionMatches | Foreign | Exception Handling | | PyErr\_Occurred | Foreign | Exception Handling | | Py\_TYPE | Foreign | Common Object Structures |   Data/control flow direction:   * Python->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: foreign function call + return + if/branching statement |

| **Case 135** <https://github.com/riverloopsec/killerbee/commit/57de78a13043dfb6deb4f479f962e2c008aa753d> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/riverloopsec/killerbee/pull/126> , which points to an earlier report at <https://github.com/riverloopsec/killerbee/pull/66> where it says    Thus, the symptom is incorrect result/output |
| **Bug location:**  Processing of mic and encrypted data: In the new implementation, mic and encrypted data are directly extracted from the pkt object, instead of first accessing the fields and then operating on them. This approach reduces the number of lines of code and improves efficiency.  Change in the calculation method for crop\_size: The new implementation directly uses the length of mic and the length of encrypted data to calculate crop\_size. This method is more direct and avoids additional access to the ZigbeeSecurityHeader layer.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/riverloopsec/killerbee/commit/57de78a13043dfb6deb4f479f962e2c008aa753d>    Bug is at line 464 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  464(py\_scapy\_extensions\_17-cfg.dot) (kbdecrypt)  …  -> 462(py\_scapy\_extensions\_17-cfg.dot) (kbdecrypt)  -> 217(c\_zigbee\_crypt\_2-cfg.dot) (zigbee\_crypt\_decrypt\_ccm)  Along the backward search path until origin, the cross-language functions involved are:     | **Function** | **Type** | **Semantics** | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * C->Python |
| **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 calculation of the variable  Location: assignment statement |

| **Case 136** <https://github.com/pygobject/pycairo/commit/590bcd2ecc9c29300e2f0225b8a6fab857b838d7> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/190> at where it says    Thus, the symptom is warning message (deprecation warnings) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygobject/pycairo/commit/590bcd2ecc9c29300e2f0225b8a6fab857b838d7>    Bug is at line 480 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  480(c\_surface\_27-cfg.dot) (surface\_set\_mime\_data)  …  -> 459(c\_surface\_27-cfg.dot) (surface\_set\_mime\_data)  -> 456(py\_test\_surface\_37-cfg.dot) (test\_surface\_get\_set\_mime\_data)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyMem\_Malloc | Foreign | Memory Management) | | PyObject\_AsReadBuffer | Foreign | Old Buffer Protocol |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API function misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change to proper API function  Location: API function + assignment statement |

| **Case 137** <https://github.com/pygobject/pycairo/commit/065a7a7aed9e5523dcbf41568d89e94fc0af44b9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/50> at where it says    Thus, the symptom is incorrect result/output. |
| **Bug location:**  Analysis  Before these lines of code were added, calling the cairo\_surface\_set\_device\_scale function to set certain specific values for the device's scale ratio could lead to assertion failures within the Cairo library. This is because Cairo internally uses a matrix to handle such scaling, and some scaling values might render this matrix non-invertible (for example, when the scaling ratio is 0). In Cairo, attempting to invert a non-invertible matrix triggers an assertion error.  Solution  To avoid such situations, the added code first initializes a scaling matrix transform using the provided x and y scaling values before setting the device scale ratio. Then, it attempts to invert this matrix using the cairo\_matrix\_invert function. If the matrix is non-invertible, this function returns an error, and the newly added RETURN\_NULL\_IF\_CAIRO\_ERROR macro catches this error and returns NULL.  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygobject/pycairo/commit/065a7a7aed9e5523dcbf41568d89e94fc0af44b9>    Bug is at line 331 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  331(c\_surface\_20-cfg.dot) (surface\_set\_device\_scale)  …  -> 323(c\_surface\_20-cfg.dot) (surface\_set\_device\_scale)  -> 69(py\_test\_surface\_9-cfg.dot) (test\_surface\_set\_device\_scale)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 variable and function call  Location: function call |

| **Case 138** <https://github.com/pygobject/pycairo/commit/42eb8a30f57676e74a194be297d051f457cebf23> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/48> which points to an earlier report at <https://bugs.freedesktop.org/show_bug.cgi?id=101866> 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/pygobject/pycairo/commit/42eb8a30f57676e74a194be297d051f457cebf23>    Bug is at line 69 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  69(c\_pattern\_1-cfg.dot) (PycairoPattern\_FromPattern)  …  -> 40(c\_pattern\_1-cfg.dot) (PycairoPattern\_FromPattern)  -> 421(c\_context\_39-cfg.dot) (pycairo\_get\_source)  -> 422(c\_context\_39-cfg.dot) (pycairo\_get\_source)  -> 419(c\_context\_39-cfg.dot) (pycairo\_get\_source)  -> 8(py\_test\_pattern\_1-cfg.dot) (test\_cmp\_hash)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 branch in switch  Location: assignment statement |

| **Case 139** <https://github.com/pygobject/pycairo/commit/65aea3dfdc5b131bc39a69ba00062f799fd72489> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/57> 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/pygobject/pycairo/commit/65aea3dfdc5b131bc39a69ba00062f799fd72489>    Bug is at line 297 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  297(c\_surface\_16-cfg.dot) (surface\_get\_device)  ...  -> 292(c\_surface\_16-cfg.dot) (surface\_get\_device)  -> 10(py\_test\_device\_1-cfg.dot) (test\_get\_device)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PycairoDevice\_FromDevice | NAX |  |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data reference error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a function call to get the reference of variable  Location: return statement + NAX function |

| **Case 140** <https://github.com/pygobject/pycairo/commit/be23eb87cf592a4ee5c616b2de3864fd8c2d5b43> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/57> 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/pygobject/pycairo/commit/be23eb87cf592a4ee5c616b2de3864fd8c2d5b43>    Bug is at line 594 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  594(c\_surface\_16-cfg.dot) (image\_surface\_create\_for\_data)  ...  -> 552(c\_surface\_16-cfg.dot) (image\_surface\_create\_for\_data)  -> 181(py\_test\_api\_11-cfg.dot) (test\_image\_surface\_create\_for\_data)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyObject\_AsWriteBuffer | Foreign | Old Buffer Protocol | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data reference error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change function call  Location: return statement + NAX function |

| **Case 141** <https://github.com/pygobject/pycairo/commit/e2154f115d8ef32221f8a1fe08bdee49d23518ba> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/8> , which points to an earlier report at <https://bugs.freedesktop.org/show_bug.cgi?id=89162> where it says    Thus, the symptom is crash or incorrect result/output. |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/pygobject/pycairo/commit/e2154f115d8ef32221f8a1fe08bdee49d23518ba>    Bug is at line 111 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  111(c\_matrix\_15-cfg.dot) (matrix\_operator\_multiply)  ...  -> 108(c\_matrix\_15-cfg.dot) (matrix\_operator\_multiply)  -> 181(py\_test\_api\_11-cfg.dot) (test\_matrix)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyObject\_IsInstance | Foreign | Object Protocol |   Data/control flow direction:   * Python->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 + exception handling  Location: if/branching statement |

| **Case 142** <https://github.com/pygobject/pycairo/commit/07ee64b3fb51a16ded90822a0fb00cb8b180b839> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/pygobject/pycairo/issues/41> 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/pygobject/pycairo/commit/07ee64b3fb51a16ded90822a0fb00cb8b180b839>    Bug is at line 783 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  783(c\_surface\_16-cfg.dot) (image\_surface\_get\_data)  ...  -> 765(c\_surface\_16-cfg.dot) (image\_surface\_get\_data)  -> 14(py\_test\_surface\_9-cfg.dot) (test\_image\_surface\_get\_data\_refcount)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyBuffer\_FillInfo | Foreign | Buffer Protocol | | PyMemoryView\_FromBuffer | Foreign | MemoryView objects |   Data/control flow direction:   * Python->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 a new function to handle the reference count  Location: return statement + foreign function |

| **Case 143** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/335e105bc1208fd3a99c419a448792dd5f95ca5d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/188> 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/adafruit/adafruit-beaglebone-io-python/commit/335e105bc1208fd3a99c419a448792dd5f95ca5d>    Bug is at line 199 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  196(c\_event\_gpio\_6-cfg.dot) (open\_value\_file)  -> 193(c\_event\_gpio\_6-cfg.dot) (open\_value\_file)  -> 190(c\_event\_gpio\_6-cfg.dot) (open\_value\_file)  -> 806(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  ...  -> 800(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  -> 129(c\_event\_gpio\_2-cfg.dot) (gpio\_export)  ...  -> 76(c\_event\_gpio\_2-cfg.dot) (gpio\_export)  -> 800(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  ...  -> 784(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  -> 478(c\_py\_gpio\_13-cfg.dot) (py\_wait\_for\_edge)  ...  -> 446(c\_py\_gpio\_13-cfg.dot) (py\_wait\_for\_edge)  -> 25(py\_button\_32-cfg.dot) (wait\_for\_edge)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Function call misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: delete the burden function call  Location: function call |

| **Case 144** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/5df30a8b735470e9b47f4f416a955043d85576a9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/310> where it says    Thus, the symptom is error message (value error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/5df30a8b735470e9b47f4f416a955043d85576a9>    Bug is at line 24 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  24(c\_c\_pinmux\_1-cfg.dot) (set\_pin\_mode)  …  -> 9(c\_c\_pinmux\_1-cfg.dot) (set\_pin\_mode)  -> 142(c\_py\_gpio\_4-cfg.dot) (py\_setup\_channel)  …  -> 74(c\_py\_gpio\_4-cfg.dot) (py\_setup\_channel)  -> 10(c\_gpio\_31-cfg.dot) (setup)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 145** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/9298393cf57e7fd9c2414cf8a736df022fecedb3> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/308> 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/adafruit/adafruit-beaglebone-io-python/commit/9298393cf57e7fd9c2414cf8a736df022fecedb3>    Bug is at line 12 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  12(c\_c\_pinmux\_1-cfg.dot) (set\_pin\_mode)  …  -> 9(c\_c\_pinmux\_1-cfg.dot) (set\_pin\_mode)  -> 142(c\_py\_gpio\_4-cfg.dot) (py\_setup\_channel)  …  -> 74(c\_py\_gpio\_4-cfg.dot) (py\_setup\_channel)  -> 10(c\_gpio\_31-cfg.dot) (setup)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: change array size  Location: variable definetion |

| **Case 146** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/794d7136b4f14e417cd5033dd4be8545b5d741b6> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/188> 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/adafruit/adafruit-beaglebone-io-python/commit/794d7136b4f14e417cd5033dd4be8545b5d741b6>    Bug is at line 112 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  111(c\_c\_pwm\_4-cfg.dot) (initialize\_pwm)  ...  -> 97(c\_c\_pwm\_4-cfg.dot) (initialize\_pwm)  -> 477(c\_c\_pwm\_8-cfg.dot) (pwm\_setup)  -> 476(c\_c\_pwm\_8-cfg.dot) (pwm\_setup)  -> 306(c\_c\_pwm\_8-cfg.dot) (pwm\_setup)  -> 589(c\_c\_pwm\_9-cfg.dot) (pwm\_start)  ...  -> 579(c\_c\_pwm\_9-cfg.dot) (pwm\_start)  -> 77(c\_py\_pwm\_2-cfg.dot) (py\_start\_channel)  …  -> 40(c\_py\_pwm\_2-cfg.dot) (py\_start\_channel)  -> 11(py\_pwm\_33-cfg.dot) (start)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 argument  Location: function call |

| **Case 147** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/7fb13f9f3d4c559b14c55721e0a51024fb12608a> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/7fb13f9f3d4c559b14c55721e0a51024fb12608a> 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/adafruit/adafruit-beaglebone-io-python/commit/7fb13f9f3d4c559b14c55721e0a51024fb12608a>    Bug is at line 198 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  198(c\_c\_pwm\_5-cfg.dot) (pwm\_set\_frequency)  …  -> 120(c\_c\_pwm\_5-cfg.dot) (pwm\_set\_frequency)  -> 199(c\_py\_pwm\_5-cfg.dot) (py\_set\_frequency)  …  -> 176(c\_py\_pwm\_5-cfg.dot) (py\_set\_frequency)  -> 13(py\_pwm\_33-cfg.dot) (set\_frequency)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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: change array size  Location: variable definetion |

| **Case 148** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/ca0a4afcdb34edda24885c7686506d8ddf2b8913> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/ca0a4afcdb34edda24885c7686506d8ddf2b8913> 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/adafruit/adafruit-beaglebone-io-python/commit/ca0a4afcdb34edda24885c7686506d8ddf2b8913>    Bug is at line 585 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  585(c\_py\_gpio\_16-cfg.dot) (initGPIO)  …  -> 570(c\_py\_gpio\_16-cfg.dot) (initGPIO)  -> 3(py\_gpio\_31-cfg.dot) (import)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | Py\_InitModule3 | Foreign | Module Initialization | | PyModule\_Create | Foreign | Module Initialization |   Data/control flow direction:   * Python->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 argument  Location: function call |

| **Case 149** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/73ea172d212d59c6159d8e3d3ad18ffc9ac61f21> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/170> where it says    Thus, the symptom is that the program is not working |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/73ea172d212d59c6159d8e3d3ad18ffc9ac61f21>    Bug is at line 584 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  584(c\_c\_pwm\_5-cfg.dot) (pwm\_start)  …  -> 559(c\_c\_pwm\_5-cfg.dot) (pwm\_start)  -> 77(c\_py\_pwm\_5-cfg.dot) (py\_start\_channel)  …  -> 40(c\_py\_pwm\_5-cfg.dot) (py\_start\_channel)  -> 11(py\_pwm\_33-cfg.dot) (start)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Error handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add error handling statement  Location: if/branching statement + function call |

| **Case 150** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/8e64f67ff0a953bf4f3f552726ad6e8f2070e038> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/178> where it says    Thus, the symptom is error message (value error). |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/8e64f67ff0a953bf4f3f552726ad6e8f2070e038>    Bug is at line 227 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  227(c\_event\_gpio\_6-cfg.dot) (open\_value\_file)  …  -> 190(c\_event\_gpio\_6-cfg.dot) (open\_value\_file)  -> 806(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  ...  -> 800(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  -> 129(c\_event\_gpio\_2-cfg.dot) (gpio\_export)  ...  -> 76(c\_event\_gpio\_2-cfg.dot) (gpio\_export)  -> 800(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  ...  -> 784(c\_event\_gpio\_28-cfg.dot) (blocking\_wait\_for\_edge)  -> 478(c\_py\_gpio\_13-cfg.dot) (py\_wait\_for\_edge)  ...  -> 446(c\_py\_gpio\_13-cfg.dot) (py\_wait\_for\_edge)  -> 25(py\_button\_32-cfg.dot) (wait\_for\_edge)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:   * Python->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 call  Location: function call |

| **Case 151** <https://github.com/adafruit/adafruit-beaglebone-io-python/commit/b65cbf8e41b444bad7c4ef6cfd4f88a30210fd78> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/adafruit/adafruit-beaglebone-io-python/issues/185> 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/adafruit/adafruit-beaglebone-io-python/commit/b65cbf8e41b444bad7c4ef6cfd4f88a30210fd78>    Bug is at line 435 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  435(c\_c\_pwm\_8-cfg.dot) (pwm\_setup)  …  -> 306(c\_c\_pwm\_8-cfg.dot) (pwm\_setup)  -> 589(c\_c\_pwm\_9-cfg.dot) (pwm\_start)  ...  -> 579(c\_c\_pwm\_9-cfg.dot) (pwm\_start)  -> 77(c\_py\_pwm\_2-cfg.dot) (py\_start\_channel)  …  -> 40(c\_py\_pwm\_2-cfg.dot) (py\_start\_channel)  -> 11(py\_pwm\_33-cfg.dot) (start)  Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  File I/O error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Add a function call  Location: function call |

| **Case 152** <https://github.com/libcsp/libcsp/commit/03cc710bf7f72ef1f687502aacad79e5fa92fd9b> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/libcsp/libcsp/issues/8> where it says    Thus, the symptom is timeout |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/libcsp/libcsp/commit/03cc710bf7f72ef1f687502aacad79e5fa92fd9b>    Bug is at line 138 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  136(c\_csp\_buffer\_6-cfg.dot) (csp\_buffer\_clone)  ...  -> 129(c\_csp\_buffer\_6-cfg.dot) (csp\_buffer\_clone)  -> 194(c\_csp\_io\_4-cfg.dot) (csp\_send\_direct)  ...  -> 84(c\_csp\_io\_4-cfg.dot) (csp\_send\_direct)  -> 390(c\_csp\_io\_12-cfg.dot) (csp\_sendto)  ...  -> 359(c\_csp\_io\_12-cfg.dot) (csp\_sendto)  -> 402(c\_csp\_io\_13-cfg.dot) (csp\_sendto\_reply)  -> 399(c\_csp\_io\_13-cfg.dot) (csp\_sendto\_reply)  -> 396(c\_csp\_io\_13-cfg.dot) (csp\_sendto\_reply)  -> 394(c\_csp\_io\_13-cfg.dot) (csp\_sendto\_reply)  -> 291(c\_csp\_service\_handler\_10-cfg.dot) (csp\_service\_handler)  ...  -> 216(c\_csp\_service\_handler\_10-cfg.dot) (csp\_service\_handler)  -> 98(c\_pycsp\_9-cfg.dot) (pycsp\_service\_handler)  ...  -> 83(c\_pycsp\_9-cfg.dot) (pycsp\_service\_handler)  -> 90(py\_python\_bindings\_example\_server\_1-cfg.dot) (csp\_server)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCapsule\_SetPointer | Foreign | Capsules | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 type of the argument  Location: function call |

| **Case 153** <https://github.com/libcsp/libcsp/commit/60cf99087d9e9851cac3736031a573bbc462c64a> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/libcsp/libcsp/issues/159> 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/libcsp/libcsp/commit/60cf99087d9e9851cac3736031a573bbc462c64a>    Bug is at line 172 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  172(c\_[sart\_windows](https://github.com/libcsp/libcsp/commit/60cf99087d9e9851cac3736031a573bbc462c64a#diff-9348ebd56bd7af2ca94660a6453dc9ccc49e58932ce7fea19b2d992a521e7601)\_16-cfg.dot) (csp\_usart\_open)  ...  -> 134(c\_[sart\_windows](https://github.com/libcsp/libcsp/commit/60cf99087d9e9851cac3736031a573bbc462c64a#diff-9348ebd56bd7af2ca94660a6453dc9ccc49e58932ce7fea19b2d992a521e7601)\_16-cfg.dot) (csp\_usart\_open)  -> 57(c\_usart\_kiss\_14-cfg.dot) (csp\_usart\_open\_and\_add\_kiss\_interface)  ...  -> 33(c\_usart\_kiss\_14-cfg.dot) (csp\_usart\_open\_and\_add\_kiss\_interface)  -> 880(c\_pycsp\_9-cfg.dot) (pycsp\_kiss\_init)  ...  ->870(c\_pycsp\_9-cfg.dot) (pycsp\_kiss\_init)  -> 119(py\_CSPHandler\_11-cfg.dot) (\_uart)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_Error | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 if statement to save value  Location: before return statement |

| **Case 154** <https://github.com/libcsp/libcsp/commit/ba15bf68a78c1397aaaa7673426ff9ccb44bad35> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/libcsp/libcsp/issues/350> 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/libcsp/libcsp/commit/ba15bf68a78c1397aaaa7673426ff9ccb44bad35>    Bug is at line 67 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  67(c\_csp\_rtable\_cidr\_26-cfg.dot) (csp\_rtable\_set\_internal)  ...  -> 59(c\_csp\_rtable\_cidr\_26-cfg.dot) (csp\_rtable\_set\_internal)  -> 101(c\_csp\_rtable\_cidr\_26-cfg.dot) (csp\_rtable\_set)  ...  -> 89(c\_csp\_rtable\_cidr\_26-cfg.dot) (csp\_rtable\_set)  -> 599(c\_pycsp\_9-cfg.dot) (pycsp\_rtable\_set)  ...  ->590(c\_pycsp\_9-cfg.dot) (pycsp\_rtable\_set)  -> 114(py\_python\_bindings\_example\_server\_1-cfg.dot) (\_\_main\_\_)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_Error | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 relational operators  Location: conditional expression |

| **Case 155** <https://github.com/libcsp/libcsp/commit/c16ce5c8ecfe14ffb7f860b4bfebd078c2153ba2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/libcsp/libcsp/issues/46> 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/libcsp/libcsp/commit/c16ce5c8ecfe14ffb7f860b4bfebd078c2153ba2>    Bug is at line 114 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  114(c\_csp\_crc32\_36-cfg.dot) (csp\_crc32\_verify)  ...  -> 102(c\_csp\_crc32\_36-cfg.dot) (csp\_crc32\_verify)  -> 65(c\_csp\_route\_37-cfg.dot) (csp\_route\_security\_check)  ...  -> 59(c\_csp\_route\_37-cfg.dot) (csp\_route\_security\_check)  -> 179(c\_csp\_route\_37-cfg.dot) (csp\_route\_work)  ...  ->109(c\_csp\_route\_37-cfg.dot) (csp\_route\_work)  -> 468(c\_pycsp\_9-cfg.dot) (csp\_task\_router)  ...  ->464(c\_pycsp\_9-cfg.dot) (csp\_task\_router)  -> 485(c\_pycsp\_9-cfg.dot) (csp\_route\_start\_task)  ...  ->474(c\_pycsp\_9-cfg.dot) (csp\_route\_start\_task)  -> 497(c\_pycsp\_9-cfg.dot) (pycsp\_route\_start\_task)  ->496(c\_pycsp\_9-cfg.dot) (pycsp\_route\_start\_task)  -> 72(py\_csp\_server\_client\_2-cfg.dot) (main)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_Error | Foreign | Exception Handling |   Data/control flow direction:  Python->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 if statement  Location: if/branching statement + assignment statement |

| **Case 156** <https://github.com/ghaering/pysqlite/commit/9b23d8d2863715e3654a8bb4a33483aa5adf40fd> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ghaering/pysqlite/issues/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/ghaering/pysqlite/commit/9b23d8d2863715e3654a8bb4a33483aa5adf40fd>    Bug is at line 1042 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1042(c\_connection\_1-cfg.dot) (pysqlite\_connection\_set\_limit)  -> 1564(py\_datastore\_sqlite\_stub\_3-cfg.dot) (csp\_crc32\_verify)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyInt\_FromLong | Foreign | Integer Objects | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Functionality missing |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add a new function  Location: function definition |

| **Case 157** <https://github.com/ghaering/pysqlite/commit/ae4c5454849f3e0d08c35ebfface95d297340a14> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/ghaering/pysqlite/issues/89> 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/ghaering/pysqlite/commit/ae4c5454849f3e0d08c35ebfface95d297340a14>    Bug is at line 183 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  183(c\_cursor\_3-cfg.dot) (\_bind\_parameter)  ...  -> 169(c\_cursor\_3-cfg.dot) (\_bind\_parameter)  -> 392(c\_cursor\_3-cfg.dot) (\_query\_execute)  ...  -> 216(c\_cursor\_3-cfg.dot) (\_query\_execute)  -> 511(c\_cursor\_3-cfg.dot) (cursor\_execute)  ->509(c\_cursor\_3-cfg.dot) (cursor\_execute)  -> 114(py\_stress\_6-cfg.dot) (getcon)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | Foreign | Foreign | Exception Handling | | PyString\_Check | Foreign | String/Bytes Objects | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyUnicode\_Check | Foreign | Unicode Objects and Codecs | | Py\_INCREF | Foreign | Reference Counting | | PyIter\_Check | Foreign | Iterator Protocol | | PyList\_New | Foreign | List Objects | | PyTuple\_New | Foreign | Tuple Objects |   Data/control flow direction:  Python->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 variable data type  Location: function call + assignment statement |

| **Case 158** <https://github.com/morse-simulator/morse/commit/4204ce1044a4d8ac615c2121ed155bc02b95bbc4> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/morse-simulator/morse/issues/508> 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/morse-simulator/morse/commit/4204ce1044a4d8ac615c2121ed155bc02b95bbc4>    Bug is at line 44 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  44(c\_velodyne\_5-cfg.dot) (Velodyne\_init)  ...  -> 35(c\_velodyne\_5-cfg.dot) (Velodyne\_init)  -> 26(py\_velodyne\_16-cfg.dot) (initialize)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_Format | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data size error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + foreign function |

| **Case 159** <https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/astrofrog/fast-histogram/issues/51> 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/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba>    Bug is at line 305 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  305(c\_[\_histogram\_core](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_9-cfg.dot) (\_histogram2d)  ...  -> 185(py\_[\_histogram\_core](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_9-cfg.dot) (\_histogram2d)  -> 127(py\_[histogram](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_6-cfg.dot) (histogram2d)  ...  -> 63(py\_[histogram](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_6-cfg.dot) (histogram2d)  -> 226(py\_velodyne\_16-cfg.dot) (test\_mixed\_strides)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_XDECREF | Foreign | Reference Counting | | Py\_DECREF | Foreign | Reference Counting | | PyArray\_FILLWBYTE | NAX | Array API |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: save data to multiple variables  Location: assignment statement |

| **Case 160** <https://github.com/astrofrog/fast-histogram/commit/e384cdf1fec4ee65a594ca4e91478eb3c98b2d18> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/astrofrog/fast-histogram/pull/21> 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/astrofrog/fast-histogram/commit/e384cdf1fec4ee65a594ca4e91478eb3c98b2d18>    Bug is at line 54 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  54(py\_[histogram](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_6-cfg.dot) (histogram1d)  -> 56(c\_[\_histogram\_core](https://github.com/astrofrog/fast-histogram/commit/82f140be8f098e00e4eef35e8f6582c8b7067fba#diff-cc82bd514007eb08beb0ca8764ddc6b8580778a7e17d4779ac780ff5b06ba6a8)\_9-cfg.dot) (\_histogram1d)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_XDECREF | Foreign | Reference Counting |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Specail case handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit  Location: if/branching statement + assignment statement |

| **Case 161** <https://github.com/bastibe/lunatic-python/commit/348716178fb29fa21d1d45facd86d2652d62b702> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/astrofrog/fast-histogram/pull/21> 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/bastibe/lunatic-python/commit/348716178fb29fa21d1d45facd86d2652d62b702>    Bug is at line 623 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  623(c\_[pythoninlua](https://github.com/bastibe/lunatic-python/commit/348716178fb29fa21d1d45facd86d2652d62b702#diff-237a3b372d5eb0042d6ac324dc98a4a809abfde80782be01580fbcc4256d5608)\_6-cfg.dot) (luaopen\_python)  …  -> 597(c\_[pythoninlua](https://github.com/bastibe/lunatic-python/commit/348716178fb29fa21d1d45facd86d2652d62b702#diff-237a3b372d5eb0042d6ac324dc98a4a809abfde80782be01580fbcc4256d5608)\_6-cfg.dot) (luaopen\_python)  -> 589(c\_luainpython\_17-cfg.dot) (PyInit\_lua)  …  -> 572(c\_luainpython\_17-cfg.dot) (PyInit\_lua)  -> 89(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyImport\_AppendInittab | Foreign | Importing Modules | | Py\_SetProgramName | Foreign | Finalization, and Threads | | Py\_IsInitialized | Foreign | Finalization, and Threads |   Data/control flow direction:  Python->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 function to config python  Location: function call + before foreign function |

| **Case 162** <https://github.com/bastibe/lunatic-python/commit/71611c696e94d5aa7f9c90d0092abb6e61ee5880> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/pull/14> 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/bastibe/lunatic-python/commit/71611c696e94d5aa7f9c90d0092abb6e61ee5880>    Bug is at line 55, 64 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  55(c\_luainpython\_6-cfg.dot) (LuaConvert)  …  -> 40(c\_luainpython\_6-cfg.dot) (LuaConvert)  -> 138(c\_luainpython\_6-cfg.dot) (LuaCall)  …  -> 100(c\_luainpython\_6-cfg.dot) (LuaCall)  -> 491(c\_luainpython\_6-cfg.dot) (Lua\_require)  …  -> 483(c\_luainpython\_6-cfg.dot) (Lua\_require)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_SetItem | Foreign | Tuple Objects | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Print | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_DelItem | Foreign | Tuple Objects |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument of foreign function API  Location: if/branching statement + foreign function |

| **Case 163** <https://github.com/bastibe/lunatic-python/commit/bf894ed520b97645298e4e6eae377b028fc04cab#diff-37883077a2d4c9e4657a6e1a425dc820d84e3f0d6179bbe6dc41173387e292b0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/pull/40> 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/bastibe/lunatic-python/commit/bf894ed520b97645298e4e6eae377b028fc04cab#diff-37883077a2d4c9e4657a6e1a425dc820d84e3f0d6179bbe6dc41173387e292b0>    Bug is at line 467 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  467(c\_luainpython\_6-cfg.dot) (Lua\_require)  …  -> 446(c\_luainpython\_6-cfg.dot) (Lua\_require)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_SetItem | Foreign | Tuple Objects | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Print | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_DelItem | Foreign | Tuple Objects |   Data/control flow direction:  Python->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: function call |

| **Case 164** <https://github.com/bastibe/lunatic-python/commit/86e931130c32b98e2ac04b6a69c784f80997030d> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/pull/12> 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/bastibe/lunatic-python/commit/86e931130c32b98e2ac04b6a69c784f80997030d>    Bug is at line 62 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  62(c\_luainpython\_6-cfg.dot) (LuaConvert)  …  -> 40(c\_luainpython\_6-cfg.dot) (LuaConvert)  -> 138(c\_luainpython\_6-cfg.dot) (LuaCall)  …  -> 100(c\_luainpython\_6-cfg.dot) (LuaCall)  -> 491(c\_luainpython\_6-cfg.dot) (Lua\_require)  …  -> 483(c\_luainpython\_6-cfg.dot) (Lua\_require)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_SetItem | Foreign | Tuple Objects | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Print | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_DelItem | Foreign | Tuple Objects |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API misuse |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change argument of foreign function API  Location: if/branching statement + foreign function |

| **Case 165** <https://github.com/bastibe/lunatic-python/commit/ccaa0870786d15057f91d3aa4fcd6294f0569f59> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/pull/50> 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/bastibe/lunatic-python/commit/ccaa0870786d15057f91d3aa4fcd6294f0569f59>    Bug is at line 87 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  87(c\_luainpython\_6-cfg.dot) (LuaConvert)  …  -> 40(c\_luainpython\_6-cfg.dot) (LuaConvert)  -> 138(c\_luainpython\_6-cfg.dot) (LuaCall)  …  -> 100(c\_luainpython\_6-cfg.dot) (LuaCall)  -> 491(c\_luainpython\_6-cfg.dot) (Lua\_require)  …  -> 483(c\_luainpython\_6-cfg.dot) (Lua\_require)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_SetItem | Foreign | Tuple Objects | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Print | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_DelItem | Foreign | Tuple Objects |   Data/control flow direction:  Python->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 argument of NAX function  Location: assignment statement + NAX function |

| **Case 166** <https://github.com/bastibe/lunatic-python/commit/cfe4ddc978c08d8aefb2d600b9922fe71dca332f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/issues/6> where it says    Thus, the symptom is segment faulst (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/bastibe/lunatic-python/commit/cfe4ddc978c08d8aefb2d600b9922fe71dca332f>    Bug is at line 232 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  232(c\_luainpython\_6-cfg.dot) (LuaObject\_setattr)  …  -> 225(c\_luainpython\_6-cfg.dot) (LuaObject\_setattr)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:  Python->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 167** <https://github.com/bastibe/lunatic-python/commit/cfe4ddc978c08d8aefb2d600b9922fe71dca332f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bastibe/lunatic-python/issues/6> 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/bastibe/lunatic-python/commit/cfe4ddc978c08d8aefb2d600b9922fe71dca332f>      Bug is at line 38 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  38(c\_luainpython\_6-cfg.dot) (LuaObject\_New)  -> 87(c\_luainpython\_6-cfg.dot) (LuaConvert)  …  -> 40(c\_luainpython\_6-cfg.dot) (LuaConvert)  -> 138(c\_luainpython\_6-cfg.dot) (LuaCall)  …  -> 100(c\_luainpython\_6-cfg.dot) (LuaCall)  -> 491(c\_luainpython\_6-cfg.dot) (Lua\_require)  …  -> 483(c\_luainpython\_6-cfg.dot) (Lua\_require)  -> 24(py\_test\_lua\_4-cfg.dot) (\_\_main\_\_)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyCallable\_Check | Foreign | Call Protocol | | PyTuple\_SetItem | Foreign | Tuple Objects | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Print | Foreign | Exception Handling | | Py\_DECREF | Foreign | Reference Counting | | PyObject\_DelItem | Foreign | Tuple Objects |   Data/control flow direction:  Python->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 global variables to arguments  Location: function definition |

| **Case 168** <https://github.com/python-lz4/python-lz4/commit/36db8c459379a8287224e95261d0896e67cbaf70> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-lz4/python-lz4/issues/19> 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/python-lz4/python-lz4/commit/36db8c459379a8287224e95261d0896e67cbaf70>    Bug is at line 206 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  206(c\_python-lz4\_4-cfg.dot) (initlz4)  …  -> 183(c\_python-lz4\_4-cfg.dot) (initlz4)  -> 1(py\_test\_lua\_27-cfg.dot) (\_\_main\_\_)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | Py\_InitModule | Foreign | Allocating Objects on the Heap | | PyModule\_Create | Foreign | Module Objects | | PyModule\_AddStringConstant | Foreign | Module Objects | | PyErr\_NewException | Foreign | Exception Handling |   Data/control flow direction:  Python->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 two foreign function  Location: foreign function |

| **Case 169** <https://github.com/python-lz4/python-lz4/commit/b6a96a17c130ddce9f99fb0cc4443377b856c7f2> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/python-lz4/python-lz4/issues/19> 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/python-lz4/python-lz4/commit/b6a96a17c130ddce9f99fb0cc4443377b856c7f2>    Bug is at line 209 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  209(c\_[python-lz4](https://github.com/python-lz4/python-lz4/commit/36db8c459379a8287224e95261d0896e67cbaf70#diff-d602aab21b91ac635d34517ded9a959b1ef0a45e424dc3023af92bf6c3182dd3)\_4-cfg.dot) (initlz4)  …  -> 183(c\_[python-lz4](https://github.com/python-lz4/python-lz4/commit/36db8c459379a8287224e95261d0896e67cbaf70#diff-d602aab21b91ac635d34517ded9a959b1ef0a45e424dc3023af92bf6c3182dd3)\_4-cfg.dot) (initlz4)  -> 1(py\_test\_lua\_27-cfg.dot) (\_\_main\_\_)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | Py\_InitModule | Foreign | Allocating Objects on the Heap | | PyModule\_Create | Foreign | Module Objects | | PyModule\_AddStringConstant | Foreign | Module Objects | | PyErr\_NewException | Foreign | Exception Handling |   Data/control flow direction:  Python->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 a foreign function  Location: foreign function |

| **Case 170** <https://github.com/spglib/spglib/commit/1945d46d7ccf569046f24087e96f58392ded260f> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/spglib/spglib/pull/267> 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/spglib/spglib/commit/1945d46d7ccf569046f24087e96f58392ded260f>    Bug is at line 439 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  439(c\_magnetic\_spacegroup\_12-cfg.dot) (get\_reference\_space\_group)  …  -> 409(c\_magnetic\_spacegroup\_12-cfg.dot) (get\_reference\_space\_group)  -> 115(c\_magnetic\_spacegroup\_12-cfg.dot) (msg\_identify\_magnetic\_space\_group\_type)  …  -> 92(c\_magnetic\_spacegroup\_12-cfg.dot) (msg\_identify\_magnetic\_space\_group\_type)  -> 1252(c\_spglib\_21-cfg.dot) (get\_magnetic\_dataset)  …  -> 1195(c\_spglib\_21-cfg.dot) (get\_magnetic\_dataset)  -> 289(c\_spglib\_21-cfg.dot) (spgms\_get\_magnetic\_dataset)  …  -> 284(c\_spglib\_21-cfg.dot) (spgms\_get\_magnetic\_dataset)  -> 518(c\_\_spglib\_29-cfg.dot) (py\_get\_magnetic\_dataset)  …  -> 488(c\_\_spglib\_29-cfg.dot) (py\_get\_magnetic\_dataset)  -> 624(py\_spglib\_35-cfg.dot) (get\_magnetic\_symmetry\_dataset)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 171** <https://github.com/spglib/spglib/commit/876fb9d2499be0fe275018c3000b188c7c6c3a75> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/spglib/spglib/pull/267> where it says    Thus, the symptom is error message or segment fault (crash) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/spglib/spglib/commit/876fb9d2499be0fe275018c3000b188c7c6c3a75>    Bug is at line 973 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  373(c\_refinement\_14-cfg.dot) (get\_Wyckoff\_positions)  …  -> 289(c\_refinement\_14-cfg.dot) (get\_Wyckoff\_positions)  -> 195(c\_refinement\_14-cfg.dot) (ref\_get\_exact\_structure\_and\_symmetry)  …  -> 136(c\_refinement\_14-cfg.dot) (ref\_get\_exact\_structure\_and\_symmetry)  -> 74(c\_determination\_21-cfg.dot) (det\_determine\_all)  …  -> 56(c\_determination\_21-cfg.dot) (det\_determine\_all)  -> 1076(c\_spglib\_21-cfg.dot) (get\_dataset)  …  -> 1044(c\_spglib\_21-cfg.dot) (get\_dataset)  -> 316(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  …  -> 312(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  -> 420(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  …  -> 396(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  -> 404(py\_spglib\_35-cfg.dot) (get\_symmetry\_dataset)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 statement |

| **Case 172** <https://github.com/spglib/spglib/commit/bc73d97fbd8bb964574e85022b5a11efd0ff8f77> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/spglib/spglib/pull/267> 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/spglib/spglib/commit/bc73d97fbd8bb964574e85022b5a11efd0ff8f77>    Bug is at line 971 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  971(c\_spacegroup\_15-cfg.dot) (change\_basis\_monocli)  …  -> 963(c\_spacegroup\_15-cfg.dot) (change\_basis\_monocli)  -> 867(c\_spacegroup\_15-cfg.dot) (search\_hall\_number)  …  -> 817(c\_spacegroup\_15-cfg.dot) (search\_hall\_number)  -> 786(c\_spacegroup\_15-cfg.dot) (iterative\_search\_hall\_number)  …  -> 771(c\_spacegroup\_15-cfg.dot) (iterative\_search\_hall\_number)  -> 724(c\_spacegroup\_15-cfg.dot) (search\_spacegroup\_with\_symmetry)  …  -> 700(c\_spacegroup\_15-cfg.dot) (search\_spacegroup\_with\_symmetry)  -> 500(c\_spacegroup\_15-cfg.dot) (spa\_search\_spacegroup)  …  -> 478(c\_spacegroup\_15-cfg.dot) (spa\_search\_spacegroup)  -> 143(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  …  -> 112(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  -> 71(c\_determination\_21-cfg.dot) (det\_determine\_all)  …  -> 56(c\_determination\_21-cfg.dot) (det\_determine\_all)  -> 1076(c\_spglib\_21-cfg.dot) (get\_dataset)  …  -> 1044(c\_spglib\_21-cfg.dot) (get\_dataset)  -> 316(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  …  -> 312(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  -> 420(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  …  -> 396(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  -> 404(py\_spglib\_35-cfg.dot) (get\_symmetry\_dataset)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 if/branching statement  Location: if/branching statement |

| **Case 173** <https://github.com/spglib/spglib/commit/36d570bc3fbeda20903bfbbb7db05e40d284a319> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/spglib/spglib/pull/267> 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/spglib/spglib/commit/36d570bc3fbeda20903bfbbb7db05e40d284a319>    Bug is at line 1782 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1782(c\_spglib\_21-cfg.dot) (get\_symmetry\_with\_site\_tensors)  …  -> 1747(c\_spglib\_21-cfg.dot) (get\_symmetry\_with\_site\_tensors)  -> 1233(c\_spglib\_21-cfg.dot) (get\_magnetic\_dataset)  …  -> 1195(c\_spglib\_21-cfg.dot) (get\_magnetic\_dataset)  -> 289(c\_spglib\_21-cfg.dot) (spgms\_get\_magnetic\_dataset)  …  -> 284(c\_spglib\_21-cfg.dot) (spgms\_get\_magnetic\_dataset)  -> 518(c\_\_spglib\_29-cfg.dot) (py\_get\_magnetic\_dataset)  …  -> 488(c\_\_spglib\_29-cfg.dot) (py\_get\_magnetic\_dataset)  -> 624(py\_spglib\_35-cfg.dot) (get\_magnetic\_symmetry\_dataset)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 174** <https://github.com/spglib/spglib/commit/9218fa14771d967a20f00764b335bc2a265790a1> |
| --- |
| **Symptoms:**  Directly associated issue report: https://github.com/spglib/spglib/pull/195where 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/spglib/spglib/commit/9218fa14771d967a20f00764b335bc2a265790a1>    Bug is at line 644 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  644(c\_overlap\_17-cfg.dot) (check\_total\_overlap\_for\_sorted)  …  -> 621(c\_overlap\_17-cfg.dot) (check\_total\_overlap\_for\_sorted)  -> 304(c\_overlap\_17-cfg.dot) (ovl\_check\_total\_overlap)  …  -> 264(c\_overlap\_17-cfg.dot) (ovl\_check\_total\_overlap)  -> 532(c\_symmetry\_18-cfg.dot) (search\_translation\_part)  …  -> 501(c\_symmetry\_18-cfg.dot) (search\_translation\_part)  -> 472(c\_symmetry\_18-cfg.dot) (get\_translation)  …  -> 439(c\_symmetry\_18-cfg.dot) (get\_translation)  -> 259(c\_symmetry\_18-cfg.dot) (sym\_get\_pure\_translation)  …  -> 250(c\_symmetry\_18-cfg.dot) (sym\_get\_pure\_translation)  -> 269(c\_primitive\_20-cfg.dot) (get\_primitive)  …  -> 250(c\_primitive\_20-cfg.dot) (get\_primitive)  -> 135(c\_primitive\_20-cfg.dot) (prm\_get\_primitive)  …  -> 133(c\_primitive\_20-cfg.dot) (prm\_get\_primitive)  -> 138(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  …  -> 112(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  -> 71(c\_determination\_21-cfg.dot) (det\_determine\_all)  …  -> 56(c\_determination\_21-cfg.dot) (det\_determine\_all)  -> 1076(c\_spglib\_21-cfg.dot) (get\_dataset)  …  -> 1044(c\_spglib\_21-cfg.dot) (get\_dataset)  -> 316(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  …  -> 312(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  -> 420(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  …  -> 396(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  -> 404(py\_spglib\_35-cfg.dot) (get\_symmetry\_dataset)        Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 175** <https://github.com/spglib/spglib/commit/c571f29d57d51ed5d8efa1eb78e42e7f48082897> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/spglib/spglib/commit/c571f29d57d51ed5d8efa1eb78e42e7f48082897> 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/spglib/spglib/commit/c571f29d57d51ed5d8efa1eb78e42e7f48082897>    Bug is at line 439 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  418(c\_spin\_37-cfg.dot) (get\_operations)  …  -> 347(c\_spin\_37-cfg.dot) (get\_operations)  -> 239(c\_symmetry\_18-cfg.dot) (sym\_get\_operation)  …  -> 235(c\_symmetry\_18-cfg.dot) (sym\_get\_operation)  -> 493(c\_spacegroup\_15-cfg.dot) (spa\_search\_spacegroup)  …  -> 481(c\_spacegroup\_15-cfg.dot) (spa\_search\_spacegroup)  -> 143(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  …  -> 112(c\_determination\_21-cfg.dot) (get\_spacegroup\_and\_primitive)  -> 71(c\_determination\_21-cfg.dot) (det\_determine\_all)  …  -> 56(c\_determination\_21-cfg.dot) (det\_determine\_all)  -> 1076(c\_spglib\_21-cfg.dot) (get\_dataset)  …  -> 1044(c\_spglib\_21-cfg.dot) (get\_dataset)  -> 316(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  …  -> 312(c\_spglib\_21-cfg.dot) (spgat\_get\_dataset\_with\_hall\_number)  -> 420(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  …  -> 396(c\_\_spglib\_29-cfg.dot) (py\_get\_dataset)  -> 404(py\_spglib\_35-cfg.dot) (get\_symmetry\_dataset)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyArray\_DIMS | NAX | Array API | | PyArray\_DATA | NAX | Array API |   Data/control flow direction:  Python->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 176** <https://github.com/bcdev/jpy/commit/81bfc112257504a3ab2cc3de3ae38ca6fa4bbf04> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/52> 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/bcdev/jpy/commit/81bfc112257504a3ab2cc3de3ae38ca6fa4bbf04>    Bug is at line 597 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  597(c\_jpy\_jobj\_6-cfg.dot) (JObj\_sq\_ass\_item)  …  -> 586(c\_jpy\_jobj\_6-cfg.dot) (JObj\_sq\_ass\_item)  -> 718(c\_jpy\_jobj\_6-cfg.dot) (JType\_InitSlots)  …  -> 669(c\_jpy\_jobj\_6-cfg.dot) (JType\_InitSlots)  -> 186(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  …  -> 133(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  -> 127(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  …  -> 65(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  -> 596(c\_jpy\_module\_7-cfg.dot) (JPy\_array)  …  -> 579(c\_jpy\_module\_7-cfg.dot) (JPy\_array)  -> 104(py\_jpy\_array\_test\_23-cfg.dot) (test\_array\_item\_del)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | JPy\_IS\_STR | NAX | String API | | PyErr\_SetString | Foreign | Exception Handling | | JPy\_IS\_CLONG | NAX | Int API |   Data/control flow direction:  Python->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 177**  <https://github.com/bcdev/jpy/commit/58f81d92b6fdf4bf2623b88a1414ba64bfc18788> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/52> 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/bcdev/jpy/commit/58f81d92b6fdf4bf2623b88a1414ba64bfc18788>    Bug is at line 114 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  114(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_MatchPyArgs)  …  -> 81(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_MatchPyArgs)  -> 638(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_FindMethod0)  …  -> 607(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_FindMethod0)  -> 696(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_FindMethod)  …  -> 670(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_FindMethod)  -> 770(c\_jpy\_jobj\_6-cfg.dot) (JObj\_init)  …  -> 92(c\_jpy\_jobj\_6-cfg.dot) (JObj\_init)  -> 718(c\_jpy\_jobj\_6-cfg.dot) (JType\_InitSlots)  …  -> 693(c\_jpy\_jobj\_6-cfg.dot) (JType\_InitSlots)  -> 186(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  …  -> 133(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  -> 127(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  …  -> 65(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  -> 532(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  …  -> 518(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  -> 99(py\_jpy\_array\_test\_23-cfg.dot) (test\_staticMethodIsFoundOverNonStatic)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 178**  <https://github.com/bcdev/jpy/commit/d183a296be7b54cbd347ed37d41ee9845428a9e9> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/64> 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/bcdev/jpy/commit/d183a296be7b54cbd347ed37d41ee9845428a9e9>    Bug is at line 1031 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1031(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassFields)  …  -> 994(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassFields)  -> 829(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  …  -> 795(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  -> 204(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  …  -> 118(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  -> 127(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  …  -> 65(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  -> 532(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  …  -> 518(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  -> 99(py\_jpy\_array\_test\_23-cfg.dot) (test\_staticMethodIsFoundOverNonStatic)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 179**  <https://github.com/bcdev/jpy/commit/8ef5c56c3609ee46687af3864cad3bf2281aafb1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/bcdev/jpy/commit/8ef5c56c3609ee46687af3864cad3bf2281aafb1>    Bug is at line 300 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  300(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_InvokeMethod)  …  -> 181(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_InvokeMethod)  -> 800(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_call)  …  -> 788(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_call)  -> 752(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_New)  …  -> 750(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_New)  -> 1266(c\_jpy\_jtype\_9-cfg.dot) (JType\_AddMethod)  …  -> 1252(c\_jpy\_jtype\_9-cfg.dot) (JType\_AddMethod)  -> 925(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessMethod)  …  -> 884(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessMethod)  -> 1004(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassConstructors)  …  -> 975(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassConstructors)  -> 837(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  …  -> 815(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  -> 221(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  …  -> 133(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  -> 127(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  …  -> 65(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  -> 532(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  …  -> 518(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  -> 91(py\_jpy\_overload\_test\_23-cfg.dot) (test\_toReproduceAndFixIssue56)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  API error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: assignment after initialization  Location: assignment statement |

| **Case 180**  <https://github.com/bcdev/jpy/commit/c304a392db88c468f0412167cf1abf284e550ed0> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/bcdev/jpy/commit/c304a392db88c468f0412167cf1abf284e550ed0>    Bug is at line 238 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  300(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_InvokeMethod)  …  -> 181(c\_jpy\_jmethod\_5-cfg.dot) (JMethod\_InvokeMethod)  -> 800(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_call)  …  -> 788(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_call)  -> 752(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_New)  …  -> 750(c\_jpy\_jmethod\_5-cfg.dot) (JOverloadedMethod\_New)  -> 1266(c\_jpy\_jtype\_9-cfg.dot) (JType\_AddMethod)  …  -> 1252(c\_jpy\_jtype\_9-cfg.dot) (JType\_AddMethod)  -> 925(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessMethod)  …  -> 884(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessMethod)  -> 1004(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassConstructors)  …  -> 975(c\_jpy\_jtype\_9-cfg.dot) (JType\_ProcessClassConstructors)  -> 837(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  …  -> 815(c\_jpy\_jtype\_9-cfg.dot) (JType\_ResolveType)  -> 221(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  …  -> 133(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetType)  -> 127(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  …  -> 65(c\_jpy\_jtype\_9-cfg.dot) (JType\_GetTypeForName)  -> 532(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  …  -> 518(c\_jpy\_module\_7-cfg.dot) (JPy\_get\_type)  -> 75(py\_jpy\_overload\_test\_23-cfg.dot) (test\_toReproduceAndFixIssue56)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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: assignment statement + function call |

| **Case 181**  <https://github.com/Nykakin/chompjs/commit/d486131848893714ca04f3fbf2227f71b471bdcc> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> where it says    Thus, the symptom is hang/dead lock |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/Nykakin/chompjs/commit/d486131848893714ca04f3fbf2227f71b471bdcc>    Bug is at line 164 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  164(c\_parser\_5-cfg.dot) (json)  …  -> 100(c\_parser\_5-cfg.dot) (json)  -> 86(c\_parser\_5-cfg.dot) (begin)  …  -> 80(c\_parser\_5-cfg.dot) (begin)  -> 66(c\_parser\_5-cfg.dot) (init\_lexer)  …  -> 58(c\_parser\_5-cfg.dot) (init\_lexer)  -> 18(c\_module\_7-cfg.dot) (parse\_python\_object)  …  -> 11(c\_module\_7-cfg.dot) (parse\_python\_object)  -> 25(py\_chompjs\_9-cfg.dot) (parse\_js\_object)      Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 182**  <https://github.com/Nykakin/chompjs/commit/ee79fe889b155e7ee1ce249b7361726254b52d0e> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/Nykakin/chompjs/commit/ee79fe889b155e7ee1ce249b7361726254b52d0e>    Bug is at line 157 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  157(c\_parser\_5-cfg.dot) (json)  …  -> 100(c\_parser\_5-cfg.dot) (json)  -> 86(c\_parser\_5-cfg.dot) (begin)  …  -> 80(c\_parser\_5-cfg.dot) (begin)  -> 66(c\_parser\_5-cfg.dot) (init\_lexer)  …  -> 58(c\_parser\_5-cfg.dot) (init\_lexer)  -> 18(c\_module\_7-cfg.dot) (parse\_python\_object)  …  -> 11(c\_module\_7-cfg.dot) (parse\_python\_object)  -> 25(py\_chompjs\_9-cfg.dot) (parse\_js\_object)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_BuildValue | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 beak in branching  Location: if/branching statement |

| **Case 183**  <https://github.com/manodeep/Corrfunc/commit/a8718cb2e9d89eaee2e236f8c2e537dbe65b1317> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/manodeep/Corrfunc/commit/a8718cb2e9d89eaee2e236f8c2e537dbe65b1317>    Bug is at line 247 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  247(c\_countpairs\_xi\_8-cfg.dot) (countpairs\_xi)  …  -> 56(c\_countpairs\_xi\_8-cfg.dot) (countpairs\_xi)  -> 2169(c\_\_countpairs\_5-cfg.dot) (countpairs\_countpairs\_xi)  …  -> 1990(c\_\_countpairs\_5-cfg.dot) (countpairs\_countpairs\_xi)  -> 25(py\_xi\_21-cfg.dot) (xi)    Along the backward search path until origin, the cross-language functions involved are:       | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_XDECREF | Foreign | Reference Counting | | PyArray\_DATA | NAX | Array API | | PyArray\_FromArray | NAX | Array API | | PyObject\_Print | Foreign | Object Protocol |   Data/control flow direction:  Python->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 order of statements  Location: assignment statement + loop |

| **Case 184**  <https://github.com/manodeep/Corrfunc/commit/f606c8bc7c7f358ac4085a4c8702d51f44bb1cf3> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/manodeep/Corrfunc/commit/f606c8bc7c7f358ac4085a4c8702d51f44bb1cf3>    Bug is at line 238 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  238(c\_countpairs\_xi\_8-cfg.dot) (countpairs\_xi)  …  -> 56(c\_countpairs\_xi\_8-cfg.dot) (countpairs\_xi)  -> 2169(c\_\_countpairs\_5-cfg.dot) (countpairs\_countpairs\_xi)  …  -> 1990(c\_\_countpairs\_5-cfg.dot) (countpairs\_countpairs\_xi)  -> 25(py\_xi\_21-cfg.dot) (xi)    Along the backward search path until origin, the cross-language functions involved are:       | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | Py\_XDECREF | Foreign | Reference Counting | | PyArray\_DATA | NAX | Array API | | PyArray\_FromArray | NAX | Array API | | PyObject\_Print | Foreign | Object Protocol |   Data/control flow direction:  Python->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 185**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/2ba224c060db935c45ac377c07fa67dae88e4552> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/bcdev/jpy/issues/56> 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/sergey-dryabzhinsky/python-zstd/commit/2ba224c060db935c45ac377c07fa67dae88e4552>    Bug is at line 160 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  160(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 137(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->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 loop statement and some assignment statements  Location: loop + assignment statement |

| **Case 186**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/issues/8> 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/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c>    Bug is at line 67 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  67(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 42(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data size error |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: change the argument in foreign function  Location: assignment statement + foreign function |

| **Case 187**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/issues/8> 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/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c>    Bug is at line 114 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  114(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 42(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Thread management |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: add the thread management statements  Location: assignment statement + function call |

| **Case 188**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/issues/8> 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/sergey-dryabzhinsky/python-zstd/commit/4cdcd8fbe28ab7e775add1db975400d022bcdc9c>    Bug is at line 107 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  107(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 42(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->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: add the exception handling statements  Location: if/branching statement + foreign function |

| **Case 189**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/8416a3658cf56122cec8b4502048e7b432999ba6> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/pull/26> 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/sergey-dryabzhinsky/python-zstd/commit/8416a3658cf56122cec8b4502048e7b432999ba6>    Bug is at line 51 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  51(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 42(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 initialization value  Location: assignment statement |

| **Case 190**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/8416a3658cf56122cec8b4502048e7b432999ba6> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/pull/26> 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/sergey-dryabzhinsky/python-zstd/commit/8416a3658cf56122cec8b4502048e7b432999ba6>    Bug is at line 64 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  64(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 42(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->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: revise comparison operators  Location: if/branching statement + foreign function |

| **Case 191**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/9aca7f27cd8c33d142593ae131c11d90ecb785fd> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/issues/94> 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/sergey-dryabzhinsky/python-zstd/commit/9aca7f27cd8c33d142593ae131c11d90ecb785fd>    Bug is at line 183 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  183(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  …  -> 137(c\_python-zstd\_3-cfg.dot) (py\_zstd\_uncompress)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->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 + foreign function |

| **Case 192**  <https://github.com/sergey-dryabzhinsky/python-zstd/commit/a542fd7c8de0c081c0e563940ece5510b71e3c35> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/sergey-dryabzhinsky/python-zstd/issues/94> 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/sergey-dryabzhinsky/python-zstd/commit/a542fd7c8de0c081c0e563940ece5510b71e3c35>    Bug is at line 96 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  96(c\_python-zstd\_3-cfg.dot) (py\_zstd\_compress\_mt)  …  -> 54(c\_python-zstd\_3-cfg.dot) (py\_zstd\_compress\_mt)  -> 25(py\_base\_13-cfg.dot) (helper\_compression\_default\_level)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyErr\_Format | Foreign | Exception Handling | | PyBytes\_FromStringAndSize | Foreign | Bytes Objects | | PyBytes\_AS\_STRING | Foreign | Bytes Objects |   Data/control flow direction:  Python->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: delete foreign function call  Location: if/branching statement + assignment statement |

| **Case 193**  <https://github.com/apple/ccs-pykerberos/commit/8da60e2cf257bb4b087922a40a315f11e506d8b1> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/apple/ccs-pykerberos/issues/49> , which points to an earlier report at <https://github.com/requests/requests-kerberos/issues/76> where it says    Thus, the symptom is error message (type error) |
| **Bug location:**  Starting from the first fix location, forward search leads to the identification of bug location:  <https://github.com/apple/ccs-pykerberos/commit/8da60e2cf257bb4b087922a40a315f11e506d8b1>    Bug is at line 153 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  153(c\_kerberos\_3-cfg.dot) (authGSSClientInit)  …  -> 136(c\_kerberos\_3-cfg.dot) (authGSSClientInit)  -> 36(py\_test\_kerberos\_15-cfg.dot) (test\_gssapi)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values |   Data/control flow direction:  Python->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 foreign function  Location: conditional expression + foreign function |

| **Case 194**  <https://github.com/esheldon/fitsio/commit/60619c7c2585b0f780acd48df020e5256a822be7> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/apple/ccs-pykerberos/issues/49> , which points to an earlier report at <https://github.com/requests/requests-kerberos/issues/76> 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/esheldon/fitsio/commit/60619c7c2585b0f780acd48df020e5256a822be7>    Bug is at line 1293 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1293(c\_fitsio\_pywrap\_9-cfg.dot) (PyFITSObject\_create\_image\_hdu)  …  -> 1222(c\_fitsio\_pywrap\_9-cfg.dot) (PyFITSObject\_create\_image\_hdu)  -> 470(py\_test\_header\_13-cfg.dot) (test\_read\_comment\_history)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArray\_Check | NAX | Array API | | PyArray\_TYPE | NAX | Array API | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:  Python->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 195**  <https://github.com/esheldon/fitsio/commit/60619c7c2585b0f780acd48df020e5256a822be7> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/apple/ccs-pykerberos/issues/49> , which points to an earlier report at <https://github.com/requests/requests-kerberos/issues/76> 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/esheldon/fitsio/commit/60619c7c2585b0f780acd48df020e5256a822be7>    Bug is at line 1293 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  1293(c\_fitsio\_pywrap\_9-cfg.dot) (PyFITSObject\_create\_image\_hdu)  …  -> 1222(c\_fitsio\_pywrap\_9-cfg.dot) (PyFITSObject\_create\_image\_hdu)  -> 470(py\_test\_header\_13-cfg.dot) (test\_read\_comment\_history)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArray\_Check | NAX | Array API | | PyArray\_TYPE | NAX | Array API | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyErr\_SetString | Foreign | Exception Handling |   Data/control flow direction:  Python->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 196**  <https://github.com/MSeifert04/iteration_utilities/commit/099d342bd36b1e9a9a650480a9c640050f025216> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/157> 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/MSeifert04/iteration_utilities/commit/099d342bd36b1e9a9a650480a9c640050f025216>    Bug is at line 135 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  135(c\_\_itemidxkey\_13-cfg.dot) (itemidxkey\_repr)  …  -> 128(c\_\_itemidxkey\_13-cfg.dot) (itemidxkey\_repr)  -> 22(py\_test\_itemidxkey\_22-cfg.dot) (test\_itemidxkey\_repr1)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyUnicode\_FromString | Foreign | Unicode Objects and Codecs) | | Py\_ReprEnter | Foreign | Exception Handling |   Data/control flow direction:  Python->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 + using foreign function  Location: if/branching statement + conditional expression |

| **Case 197**  <https://github.com/MSeifert04/iteration_utilities/commit/099d342bd36b1e9a9a650480a9c640050f025216> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/157> 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/MSeifert04/iteration_utilities/commit/099d342bd36b1e9a9a650480a9c640050f025216>    Bug is at line 155 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  155(c\_\_seen\_12-cfg.dot) (seen\_repr)  …  -> 151(c\_\_seen\_12-cfg.dot) (seen\_repr)  -> 201(py\_test\_seen\_23-cfg.dot) (test\_seen\_repr0)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyUnicode\_FromString | Foreign | Unicode Objects and Codecs) | | Py\_ReprEnter | Foreign | Exception Handling |   Data/control flow direction:  Python->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 + using foreign function  Location: if/branching statement + conditional expression |

| **Case 198**  <https://github.com/MSeifert04/iteration_utilities/commit/09cfa19e4cdb3f37a1489c90fdc5a597d1b7db30> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/MSeifert04/iteration_utilities/pull/176> 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/MSeifert04/iteration_utilities/commit/09cfa19e4cdb3f37a1489c90fdc5a597d1b7db30>    Bug is at line 296 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  296(c\_deepflatten\_12-cfg.dot) (deepflatten\_setstate)  …  -> 281(c\_deepflatten\_12-cfg.dot) (deepflatten\_setstate)  -> 221(py\_test\_deepflatten\_23-cfg.dot) (test\_deepflatten\_failure10)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyArg\_ParseTuple | Foreign | Parsing arguments and building values | | PyList\_CheckExact | Foreign | List Objects | | PyList\_GET\_SIZE | Foreign | List Objects | | PyErr\_Format | Foreign | Exception Handling |   Data/control flow direction:  Python->C |
| **Root cause:**  By analyzing the forward search path (from bug fix location), the root cause is:  Data handling |
| **Fixing strategy:**  By looking at the fixed code lines,  Strategy: Boundary check against value range/limit + data value checking  Location: if/branching statement + loop + foreign function |

| **Case 199**  <https://github.com/MSeifert04/iteration_utilities/commit/09cfa19e4cdb3f37a1489c90fdc5a597d1b7db30> |
| --- |
| **Symptoms:**  Directly associated issue report:<https://github.com/MSeifert04/iteration_utilities/pull/192> 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/MSeifert04/iteration_utilities/commit/09cfa19e4cdb3f37a1489c90fdc5a597d1b7db30>    Bug is at line 171 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  171(c\_deepflatten\_12-cfg.dot) (deepflatten\_next)  …  -> 118(c\_deepflatten\_12-cfg.dot) (deepflatten\_next)  -> 216(py\_test\_deepflatten\_23-cfg.dot) (test\_deepflatten\_failure9)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyErr\_Occurred | Foreign | Exception Handling | | Py\_INCREF | Foreign | Reference Counting | | Py\_DECREF | Foreign | Reference Counting | | PyList\_GET\_SIZE | Foreign | List Objects | | PyObject\_IsInstance | Foreign | Object Protocol |   Data/control flow direction:  Python->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: Boundary check against value range/limit + data value checking  Location: if/branching statement + foreign function |

| **Case 200**  <https://github.com/MSeifert04/iteration_utilities/commit/20c9b633321168e03a8ba761fe4994fc02b8c173> |
| --- |
| **Symptoms:**  Directly associated issue report: <https://github.com/MSeifert04/iteration_utilities/pull/232> 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/MSeifert04/iteration_utilities/commit/20c9b633321168e03a8ba761fe4994fc02b8c173>    Bug is at line 43 |
| **Bug manifestation:**  From the bug location, backward search until origin in Python:  path:  43(c\_minmax\_15-cfg.dot) (PyIU\_MinMax)  …  -> 5(c\_minmax\_15-cfg.dot) (PyIU\_MinMax)  -> 21(py\_test\_minmax\_27-cfg.dot) (test\_minmax\_normal1)    Along the backward search path until origin, the cross-language functions involved are:     | Function | Type | Semantics | | --- | --- | --- | | PyTuple\_GET\_SIZE | Foreign | Tuple Objects | | PyArg\_ParseTupleAndKeywords | Foreign | Parsing arguments and building values | | PyArg\_UnpackTuple | Foreign | Parsing arguments and building values | | Py\_XINCREF | Foreign | Reference Counting | | PyErr\_SetString | Foreign | Exception Handling | | PyTuple\_New | Foreign | Tuple Objects | | PyErr\_Occurred | Foreign | Exception Handling | | PyErr\_ExceptionMatches | Foreign | Exception Handling |   Data/control flow direction:  Python->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 foreign function  Location: if/branching statement + foreign function |