Indexes

| **Number** | **Abbreviation** | **Description** |
| --- | --- | --- |
| 1 | FTCC | First TC crashes |
| 2 | NSEF | No support for external C library functions (except for C file handling functions) |
| 3 | NSFF | No support for external file-handling functions |
| 4 | NSFP | No support for function pointers |
| 5 | NSGP | No support for global void pointer |
| 6 | NSLS | No support for local static variable |
| 7 | NSSF | No support for symbolic file |
| 8 | NSSP | No support for symbolic pointers |
| 9 | RT1 | Timeout1 is reached (e.g., due to the unrealistic symbolic input values / too many symbolic variables) |
| 10 | TIWC | Target program’s Illegal write to the CROWN’s data structure that causes failure |
| 11 | UB | Existence of unreachable branches in unit-testing |
| 12 | USV | Uncovered branch caused by the unrealistic symbolic input values |
| 13 | WDFS | Weakness of dfs |

| **Reason** | **ID** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FTCC | [Sed-26](#ouldt3iem77g) |  |  |  |  |  |  |
| NSEF | [Sed-1](#xpkzjquyhogh) | [Sed-6](#qqdybicf5lhg) | [Sed-13](#a02r9kijv63) | [Sed-15](#sagj2u6kh7mq) | [Sed-33](#5c03rjcipw49) | [Sed-45](#4uh9surah05s) |  |
| NSFF | [Sed-4](#8ccmigblle0m) | [Sed-16](#j0eil6xxug5t) | [Sed-17](#8h6xmf1gfie4) | [Sed-24](#ocl5bun1tdm) | [Sed-25](#qxtyi5meaze0) | [Sed-29](#8a5q1s68mwfm) | [Sed-30](#l9k4u1p9koez) |
| NSLS | [Sed-32](#iyjadlvddslf) |  |  |  |  |  |  |
| NSSF | [Sed-3](#3vrama5n7lnc) | [Sed-12](#2769tx7sfgag) | [Sed-21](#l3aqoxa25vl4) |  |  |  |  |
| NSSP | [Sed-5](#vjo7sr5v8ysd) | [Sed-7](#7o7u555zu8v1) | [Sed-8](#1611wjsxfgdc) | [Sed-9](#a8tx7qoy2rub) | [Sed-10](#ek6weqwxd5gf) | [Sed-11](#f36bdqt0m766) | [Sed-14](#hk1802th4ctw) |
| [Sed-22](#lfz9p472iqy2) | [Sed-34](#5klqqrqbldv) | [Sed-35](#l5ra6aoj9p5j) | [Sed-36](#vezvb149kmn6) | [Sed-40](#wyrami7ev8su) | [Sed-41](#gp8zfbu8xjuo) | [Sed-44](#d889419c40f) |
| [Sed-46](#qkd3p02n02f8) |  |  |  |  |  |  |
| TIWC | [Sed-23](#3v35kiw380e6) | [Sed-37](#dcnaln1dtk1s) | [Sed-42](#c4tqma4hm322) |  |  |  |  |
| UB | [Sed-33](#5c03rjcipw49) | [Sed-43](#ajz3d8fpv4mg) | [Sed-45](#4uh9surah05s) |  |  |  |  |
| USV | [Sed-2](#6gefnk8vo23e) | [Sed-18](#aryhzhgn0qly) | [Sed-19](#bn9gny2cdmjp) | [Sed-20](#7brd6oxar4xd) | [Sed-27](#a7nq15ewhl5g) | [Sed-28](#c5at1eq15zhw) | [Sed-31](#ikb599oxmjxi) |
| [Sed-38](#t8eo24ojgyc8) | [Sed-39](#cjk5ans4szdw) |  |  |  |  |  |

Sed-1

Function Name: main

Reason: NSEF

Line No: 395-432, 433-437, 438

Description:

Step 1. Line 395: getopt\_long is an external function and the execution of this function returns EOF, the “then” branch cannot be covered by CROWN

395. **while**((opt=getopt\_long(...)) != EOF){

**... /**

**e\_strings = "\n"; \* not covered**

**compiled = 1; \***

**... /**

432. }

Step 2. Line 433: e\_strings is a local pointer variable (assigned with NULL), and its value is updated in step 1 only, the "then" branch of "if" statement cannot be covered

384. char \* e\_strings = NULL

...

433. **if** (e\_strings){

**... // not covered**

437. }

Step 3. Line 438: similar to step 2, compiled is a local integer variable (assigned with 0), and its value is updated in step 1 only, the "else" branch of "if" statement cannot be covered.

385. int compiled = 0;

...

438. **if**(!compiled){...}

else **{...} // not covered**

Sed-2

Function Name: main

Reason: USV

Line No: 442-486

Description: CR2 cannot generate more than 2 TCs due to the crash at the 2nd TC and the 1st TC contains only 1 constraint (i.e., optind == argc at Line 440) and, thus, the branch coverage is limited.

1. Step 1. optind and argc are symbolic integer variables, and their initial value is 0
2. Step 2.
   * SPF obtained: optind==argc at line 440 (optind = 0, argc = 0)
   * Execution: usage(4) at line 441 calls exit and terminates the program
3. Step 3.
   * SPF: optind !=argc (optind = **1**, argc = 0)
   * Execution: **crashes** (i.e., out of bound memory access (try to access argv[1]) ) happens because argv is a pointer parameter and CR2 allocates only **one** element (i.e., argv= malloc(**1\***sizeof(char\*\*)) )
   * Result: line 443-486 cannot be covered

440. **if** (optind == argc)

441. usage (4);

442.**compile\_string (argv[optind++]);** // equivalent to access argv[1]

**443-486. ... // not covered**

Sed-3

Function Name: close\_files

Reason: NSSF

Line No: 497

Description: CR2 assigns NULL to the pointer of FILE structure (i.e., phile = NULL), which makes line 497 cannot be executed

1. Line 496: Variable phile is a pointer to FILE structure, and it is assigned with **NULL** by the driver code of this function, so the “then” branch of line 496 cannot be covered.

496. **if** (phile) // phile = NULL

497. **fclose (phile); // not covered**

Sed-4

Function Name: compile\_file

Reason: NSFF

Line No: 531-538

Description:An external function is called and its return value (**non-symbolic**) is used by line 530, where a “if” statement is executed, and the “then” branch cannot be covered.

1. Line 529: Function getc is an external function, and its return value (non-symbolic) is assigned to variable ch
2. Line 530: ch is not symbolic variable and its value is not '#'
   1. The “then branch” of line 530 (i.e., 531-538) cannot be covered as all the variables used by this line are non-symbolic variables.

529. int ch = getc(...);

530. **if**(ch == '#')

**531-538. {...} // not covered**

Sed-5

Function Name: compile\_program

Reason: NSSP

Line No: 604-613

Description:

1. Line 602: The return value of function inchar is non-symbolic and it is assigned to variable ch
2. Line 603: ch is not symbolic variable and its value is not ','
   1. The “then” branch of line 603 (i.e., line 604-613) cannot be covered.

602. int ch = inchar();

603. **if**(ch == ',')

**604-613. {...} // not covered**

Sed-6

Function Name: compile\_program

Reason: NSEF

Line No: 618-620

Description:

1. Step1. line 552: addr\_type is a local integer variable and it is assigned with 0
2. Step2. line 600: addr\_type is assigned with 1 in the “then” branch of “if” statement **only**
   1. But the “then” branch is not covered and isdigit is an external function
3. Step3. line 617: the value of addr\_type is 0, so the “then” branch (i.e., line 618-620) cannot be covered

// Function compile\_program

struct vector\* compile\_program(...){

552. **int** addr\_type=0; // local variable

… // addr\_type is not modified

600. **if**(isdigit(...)) //line 964 of compile\_address

**{... addr\_type = 1; ...}** **// not covered**

… // addr\_type is not modified

617. **if**(addr\_type==1)

**618-620. {…} // not covered**

…

}

Sed-7

Function Name: compile\_program

Reason: NSSP

Line No: 624

Description: No symbolic variables are used by the “if” statement at line 623, which makes CROWN explore the “then” or “else” branch only.

1. Line 622. Function inchar() returns a **non-symbolic** variable and its return value is assigned to variable ch
2. Line 623. ch is non-symbolic and its value is not EOF (=-1), the “then” branch (i.e., line 624) cannot be covered by CROWN

622. int ch = inchar();

623. **if**(ch==EOF)

**624. bad\_prog(NO\_COMMAND); // not covered**

Sed-8

Function Name: compile\_program

Reason: NSSP

Line No: 644

Description: refer to Sed-5

Sed-9

Function Name: compile\_program

Reason: NSSP

Line No: 658-673

Description:refer to Sed-5

Sed-10

Function Name: compile\_program

Reason: NSSP

Line No: 754-769, 771-775, 785, 790-834, 853, 866

Description: refer to Sed-5

Sed-11

Function Name: compile\_program

Reason: NSSP

Line No: 857-864

Description:

1. Step 1. Line 850, variable length is set to 0
2. Step 2. Line 852, the return value of function inchar is not symbolic and the condition of “while” statement is not satisfied.
   1. The “then” branch of “while” statement (i.e., line 853) is not covered, where length is modified only
3. Step 3. Line 856, the value of length (=0) is assigned to num,which makes the “then” branch of “for” statement (line 857-864) unexecuted

850. int length = 0;

... // length is **not modified**

852. **while**((ch=inchar()) !=EOF && ch !=0)

853. length++; // length is modified at this line only

... // length is **not modified**

856. **for**(int num=length; **num!=0**; --num) // num == 0

**857-864.{...} // not covered**

Sed-12

Function Name: savchar

Reason: NSSF

Line No: 937

Description: CR2 assigns NULL to FILE pointer, which makes line 937 unexecuted.

1. Line 936: variable is prog\_file is a pointer to structure FILE, and it is assigned with NULL by the driver function (i.e., savchar\_driver).
2. Line 936: the “then” branch of “if” statement (i.e., line 937) cannot be covered

savchar\_driver(){

…

prog\_file = **NULL**;

…

}

929. int savchar(){

… // prog\_file is not modified

936. **if** (prog\_file)

937. **ungetc (ch, prog\_file); // not covered**

**...**

**}**

Sed-13

Function Name: compile\_address

Reason: NSEF

Line No: 957-967

Description:

1. Step 1. Line 956: variable ch is a symbolic integer variable and its initial value is 0, which makes function isdigit return false
2. Step 2. Line 956: isdigit is an **external** function, so the “then” branch of the “if” statement (i.e., line 957-967) cannot be covered

956. **if**(isdigit(ch)) // check whether ch is in range [48, 57]

**957-967. {...} // not covered**

Sed-14

Function Name: compile\_filename

Reason: NSSP

Line No: 1156

Description:

1. line 1155: name is assigned with **non-null** memory space (1 byte) in the driver function
   1. CROWN does not support symbolic pointer (i.e., name cannot be altered to NULL by CROWN), so the “then” branch of line 1155 cannot be covered

// Driver begin

compile\_filename\_driver(){

...

name = **malloc**(1\***sizeof**(**char**));

SYM\_char(name[0]);

...

}

// Driver end

1138. FILE\* compile\_filename(){

... // name is not modified

1155. **if**(name)

**1156. {...} // not covered**

**...**

1183. }

Sed-15

Function Name: compile\_filename

Reason: NSEF

Line No: 1165-1181

Description:

1. Line 1157: Function strcmp is an external function and it returns 0, which makes CROWN cover the “then” branch.
   1. Inside the “then” branch, execution is terminated as the “return” statement is executed
2. CROWN cannot alter name to make the external function strcmp return non-zero value, so lines 1165-1181 are not covered.

1157. **if**(!strcmp(name, "")){... **return**;}

**1165-1181: ... // not covered**

Sed-16

Function Name: read\_file

Reason: NSFF

Line No: 1196

Description:

1. Step 1. Line 1195: name is a symbolic string which contains only 1 byte (i.e. name == "")
2. Step 2. Line 1195: fopen returns NULL, and NULL is assigned to input\_file
3. Step 3. Line 1196: The “else” branch is not covered as CROWN can not alter name to make the external function fopen return non-NULL value

1195. input\_file = fopen(name, "r"); // name = ""

1196. **if** (input\_file==0){...}

**else {...}// else branch is not uncovered**

Sed-17

Function Name: read\_file

Reason: NSFF

Line No: 1204-1215

Description:

1. Line 1203: Function feof is an external function and returns non-zero value.
2. Line 1204-1215: The “then” branch of line 1203 cannot be covered as CROWN cannot make the external function feof return 0

1203. **while**(!feof(...))

**1204-1215. {...} // then branch is not covered**

Sed-18

Function Name: chr\_copy

Reason: USV

Line No: 1237

Description: CR2 generates unrealistic values for symbolic variables, which makes the execution crash.

1. Line 1236: len is a symbolic integer variable and its initial value is 0
   1. Step 1. TC1: len = 0, line 1237 is not executed
   2. Step 2. TC2: len =-1, which makes line 1237 crash
      1. Because CR2 only allocates 1 byte for character pointers (e.g., dest and src)

1236. **while** (len--)

1237. \*dest++ = \*src++; // **crash** , as this line can access dest[10000]

Sed-19

Function Name: execute\_program

Reason: USV

Line No: 1324-1328

Description:

1. Line 1322: The external function realloc returns NULL
   1. The “then” branch of “if” statement is covered, and the execution is terminated by calling function exit
2. Line 1324-1328 cannot be covered as CROWN cannot alter symbolic variables size and ret to make external function realloc return non-NULL value.

… // size and ret is not used by other conditions

1322. if((ret=realloc(ret, size))==NULL) {exit(4);} // ck\_realloc()

1324-1328. ... // **not covered**

Sed-20

Function Name: execute\_program

Reason: USV

Line No: 1405-1466

Description:

1. Refer to Sed-18

Sed-21

Function Name: execute\_program

Reason: NSSF

Line No: 1472-1487

Description:

1. Step 1: Line 1473: input\_file is a FILE pointer, and CR2 assigns NULL to input\_file
2. Step 2: Line 1473. Function feof tries to access the elements of input\_file, which causes crash (i.e., dereference NULL pointer) and cannot make further test inputs to cover then branch @ 1473 because no previously executed statements update input\_file
   1. The coverage information of case ‘n’ (i.e., line 1473-1479) cannot be recorded by gcov.
3. Line 1480: case ‘N’ has the same problem

1293. **switch**(cmd){

...

**1472. case 'n': // not covered begins**

**1473. if(feof(input\_file)){...} // crashed**

**...**

**1479. break; // not covered ends**

**1480. case 'N': // not covered begins**

**1481. if(feof(input\_file)){...} // crashed**

**...**

**1487. break; // Not covered ends**

… // other case statements

1685.} // End of switch statement

Sed-22

Function Name: execute\_program

Reason: NSSP

Line No: 1679

Description:

1. Line 1678: p,e, and text are pointer type (char \*)
2. Line 1678: length is a symbolic integer, and its initial value is 0, which makes p and e point to the same address (i.e., p==e).
3. Line 1678: CROWN cannot alter p and e to satisfy the condition p<e
   1. The “then” branch of line 1678 (i.e., line 1679) cannot be covered

1678. **for** (p =text, e = p + length; **p < e**; p++)

**1679. {...} // not covered**

Sed-23

Function Name: match\_address

Reason: TIWC

Line No: 1705-1714

Description: TC itself does not cause crash, but crash happens when writing symbolic information (e.g., SPF) into disk (i.e., file szd\_execution) due to the memory conflict between CROWN engine and TC.

1. Error Message:
   1. Program received signal SIGSEGV, Segmentation fault.

0x000000000054e1b0 in crown::SymbolicExecutionWriter::Serialize (this=0x58dce8, os=..., tracker=0x58dc20)

at libcrown/symbolic\_execution\_writer.cc:69

69 exprs\_[i->first]->Serialize(os);

Sed-24

Function Name: read\_pattern\_space

Reason: NSFF

Line No: 1739

Description:

1. Line 1738: feof is an external function, and returns 0.
   1. The “then” branch of line 1738 cannot be covered as CROWN cannot make external function feof return non-zero value

1738. **if**(feof(...))

**1739. return 0; // not covered**

Sed-25

Function Name: read\_pattern\_space

Reason: NSFF

Line No: 1758-1776

Description:

1. Refer to Sed-4

Sed-26

Function Name: append\_pattern\_space

Reason: FTCC

Line No: 1788-1826

Description:

1. The first TC crashes when executing line 1795
   1. Line 1795: input\_file is a FILE pointer and it is assigned with NULL, which makes function getc crash
      1. getc tries to access \*input\_file (equivalent to \*NULL)

1795. ch = getc(input\_file);

Sed-27

Function Name: line\_append

Reason: USV

Line No: 1853-1854

Description: CR2 generates unrealistic value for symbolic variable, which makes the execution crash

1. Step 1. Line 1856: length is a symbolic variable, and to satisfy the condition (length > 0), CROWN outputs : length = 21474834647
2. Step 2.bcopy (=memcpy) crashes
   1. memcpy (char \*, char \*, int) would crash if the value of the third parameter is greater than the # of bytes of the second parameter
      1. to\_text contains 1 byte
      2. from\_text contains 1 byte
      3. length = 21474834647, crash happens when trying to copying the corresponding number of bytes from from\_text to to\_text

1856. if(length>0) {...} **// not covered**

bcopy (to\_text, from\_text, length); // not covered

Sed-28

Function Name: str\_append

Reason: USV

Line No: 1871-1872

Description:

1. Refer to Sed-27

Sed-29

Function Name: ck\_fopen

Reason: NSFF

Line No: 2018-2036

Description:

1. Refer to Sed-16

Sed-30

Function Name: ck\_fclose

Reason: NSFF

Line No: 2056

Description:

1. Line 2055: fclose is an external function, and its return value is not EOF (=-1)
2. Line 2056: the “then” branch of line 2055 cannot be covered as CR2 cannot make external function fclose return EOF

2055. **if**(fclose(stream)==EOF)

2056.  **panic("Couldn't close %s",\_\_fp\_name(stream)); // not covered**

Sed-31

Function Name: add\_buffer

Reason: USV

Line No: 2175

Description:

1. Refer to Sed-18

Sed-32

Function Name: init\_syntax\_once

Reason: NSLS

Line No: 2302

Description:

1. Step 1. Line 2299: done is a local static variable and it is assigned with 0
2. Step 2. Line 2301: the “then” branch cannot be reached as the value of done is 0
3. Line 2317: done is set to 1, which means the second call of the target function (i.e., init\_syntax\_once) can cover line 2302.
   1. CR2 performs unit-level testing (i.e., each TC can execute the target function only once) and the target function is not a recursive function.
      1. So line 2302 cannot be covered.

void init\_syntax\_once(){

...

2299. **static** **int** done = 0;

... // done is not modified

2301. **if**(done)

**2302. return; // not covered**

...

2317. done = 1;

...

}

Sed-33

Function Name: regex\_compile

Reason: NSEF, UB

Line No: 3334,3369

Description:

1. Step 1. Line 3332: stack = malloc(640)
2. Step 2. Line 3333: the return value of malloc is not NULL when memory is enough
   1. The machine used by CR2 has 16 GB (>> 640 B) memory
3. Step 3. Line 3334: the “then” branch of line 3333 cannot be covered

3332. void\* stack = malloc(640);

3333 **if**(stack !=NULL)

3334.  **return 12; // not covered**

Sed-34

Function Name: regex\_compile

Reason: NSSP

Line No: 3378-4231

Description:

1. Line 3285: const char\* p = pattern, \*pend = pattern+size
2. size is symbolic integer variable and its initial value is 0
   1. So p==pend
3. Line 3377: the “then” branch of this line cannot be covered as CR2 cannot alter p and pend to satisfy p!=pend

3285. **const** **char**\* p = pattern, \*pend = pattern+size;

... // p ,pend and size is not altered

3377. **while**(p!=pend)

**3378-4231{...} // not covered**

... // p ,pend and size is not altered

Sed-35

Function Name: at\_beginline\_loc\_p

Reason: NSSP

Line No: 4338

Description:

1. p is a pointer which points to a **1-byte** symbolic character variable
2. Step 1. Line 4337: prev = p-2, so \*prev (i.e., p[-2]) is not a symbolic variable
3. Step 2. Line 4338: prev and pattern are pointer variable, and prev[-1] (i.e., p[-3]) is not symbolic variable
   1. 2 branches at line 4338 are not covered as no symbolic variables are used by the conditions at this line.

4337. **const** **char** \* prev = p-2;

4338. **bool** prev\_prev\_backslash = prev > pattern && prev[-1] == '\\';

**(“else” not covered)** **(“then” not covered)**

Sed-36

Function Name: compile\_range

Reason: NSSP

Line No: 4413, 4432

Description:

1. Refer to Sed-34

Sed-37

Function Name: re\_compile\_fastmap

Reason: TIWC

Line No: 4734-5002

Description:

1. Refer to Sed-23

Sed-38

Function Name: re\_match\_2

Reason: USV

Line No: 5734-6483

Description:

1. Line 5721: \*p is a symbolic integer, and its initial value is 0;
   1. TC1: case ‘0’ is reached, and execution exits normally.

TC2: the case “default” is reached, and crash happens (as function abort is called)

* + 1. other “case” statements of the “switch” statement cannot be covered

5363. int re\_match\_2(...){

…

5721. **switch** (\*p++){ // \*p is a symbolic integer

**case** 0: **break**; // execution of TC1

**case 1: ... break; // not covered**

**case 2; ... break; // not covered**

**... // not covered**

**default**: **abort();** // execution of TC2, crash happens

6484. }

…

return -1

}

Sed-39

Function Name: re\_match\_2

Reason: USV

Line No: 5591-5612

Reason: Refer to Sed-38

Sed-40

Function Name: group\_match\_null\_string\_p

Reason: NSSP

Line No: 6573-6651

Description: No symbolic variables are used by the branch condition at line 6573.

1. Step 1. Line 6565: p is a symbolic pointer variable (char\*\*), and CR2 only allocates 1 element for this pointer (i.e., only p[0][0] is symbolic variable)
   1. \*p1 is assigned with p[0][2], which is **not a symbolic variable**
2. Step 2. Only the “default” case of the switch statement can be covered

6565. unsigned **char** \*p1 = \*p+2; // \*p1 is not a symbolic variable

… // \*p1 is not altered

6573. **switch** (\*p1){ // \*p1 is a not symbolic

**case 14: ... break; // not covered**

**case 6: ... break; // not covered**

**default**: ... **break**;

6656. }

Sed-41

Function Name: alt\_match\_null\_string\_p

Reason: NSSP

Line No: 6675

Description:

1. Line 6675: p and pend are symbolic pointer, CROWN cannot alter p and pend to satisfy the “else” branch of the while statement

… // p < pend

6675. **while**(p<pend) {...}

**{...} // else branch is not covered**

Sed-42

Function Name: regexec

Reason: TIWC

Line No: 7038-7042, 7053-7067

Description: First TC itself does not crash, but CROWN engine crashes when it tries to write the symbolic information (e.g., SPF) into the physical files (i.e., szd\_execution)

Error message: 0x00007ffff74a2c82 in \_\_GI\_\_\_assert\_fail (assertion=0x562e51 "stack\_.size() > 0",

file=0x562e30 "libcrown/symbolic\_interpreter.cc", line=140

Sed-43

Function Name: regerror

Reason: UB

Line No: 7094

Description:

1. Line 7094: An explicit crash function (i.e., abort) is called, the coverage of this line is not recorded by gcov

7093. **if**(cond) // cond is the condition of the if statement

7094.  **abort(); // not recorded**

Sed-44

Function Name: regerror

Reason: NSSP

Line No: 7100

Description:

1. Line 7099: pointer variable msg is a **non-NULL** symbolic pointer (CR2 allocates one byte for this variable)
2. Line 7100: CROWN cannot alter msg to NULL, which makes this line unreachable

regerror\_driver(){

…

msg = malloc(1\*sizeof(char));

…

}

7078. size\_t regerror(...){

… // msg is not altered

7099. **if**(!msg)

**7100. msg = "Success"; // not covered**

…

7116.}

Sed-45

Function Name: regerror

Reason: NSEF, UB

Line No: 7108-7109

Description: CR2 cannot explore the branch as the SPF of the branch is UNSAT (i.e. unsolvable constraints)

Step 1. errbuf\_size is an **unsigned** symbolic integer

Step 2. Line 7102: msg\_size = 1 as the external function strlen returns 0

Step 3. Line 7104: errbuf\_size != 0

Step 4. Line 7106: if(msg\_size > errbuf\_size)

* 1. The SPF of “then” branch of this condition is :
     1. errbuf\_size !=0(line 7104) && errbuf\_size < 1(line 7106) (UNSAT for **unsigned** **integer** type)
     2. The “then” branch cannot be covered.

7102. unsigned msg\_size = strlen(msg) + 1; // msg\_size = 1

7103.

7104. **if**(errbuf\_size !=0){

7105.

7106. **if**(msg\_size > errbuf\_size)

7108-7109. **{...} // not covered (errbuf\_size!=0 && errbuf\_size < 1) is UNSAT**

**else**{...}

7113. } // end of if statement at line 7104

Sed-46

Function Name: regfree

Reason: NSSP

Line No: 7125, 7132, 7138

Description:

1. Refer to Sed-44