|  |  |  |  |
| --- | --- | --- | --- |
| Invalid memory accesses | int i, t[10], \*p;  void main()  {  for (i=0; i<=10; i++)  if (unknownfun()) t[i] = i;  p = t + 12;  if (unknownfun()) \*p = i;  p[-6] = i;  } | In the above example, the analysis is not able to guarantee that the memory accesses t[i] and \*p are valid, so it emits a proof obligation for each:  invalid.c:5: ... accessing out of bounds index. assert i < 10; invalid.c:7: ... out of bounds write. assert \valid(p); | CWE-119  CWE-786  CWE-788 |
| Division by zero | int A, B;  void main(int x, int y)  {  A = 100 / (x \* y);  B = 333 % x;  } | div.c:4: ... division by zero. assert (int)(x\*y) != 0;  div.c:5: ... division by zero. assert x != 0; | CWE-369 |
| Undefined logical shift | void main(int c){  int x;  c = c ? 1 : 8 \* sizeof(int);  x = 1 << c;  } | shift .c:4: ... invalid RHS operand for shift. assert 0 <= c < 32; |  |
| left-shifting a negative integer | void main(int c){  int x;  x = -7,  x = x << 2;  } | lshift.c:4:[kernel] warning: invalid LHS operand for left shift. assert 0 <= x; |  |
| Overflow in integer arithmetic |  |  | CWE-190 |
| Overflow in conversion from floating-point to integer | #include "\_\_fc\_builtin.h"  int main()  {  float f = Frama\_C\_float\_interval(2e9, 3e9);  return (int) f;  } | ov\_float\_int.c:6:[kernel] warning: overflow in conversion from floating-point to integer. assert -2147483649 < f < 2147483648; [value] Values at end of function main: f ∈ [2000000000. .. 3000000000.] \_\_retres ∈ [2000000000..2147483647] | CWE-681 |
| Floating-point alarms | double sum(double a, double b)  {  return a+b;  } | double\_op\_res.c:3:[kernel] warning: non-finite double value assert \ is\_finite ((double)(a+b)); |  |
| Uninitialized variables and dangling pointers to local variables | int \*f(int c) {  int r, t,\*p;  if (c) r =2;  t = r + 3;  return &t;  }  int main(int c) {  int \*p;  p = f(c);  return \*p;  } | uninitialized.c:5: ... accessing uninitialized left-value. assert \initialized(&r);  uninitialized.c:13: ... accessing left-value that contains escaping addresses.assert !\dangling(&p); | CWE-457 |
| Undefined pointer comparison alarms | int x,y,\*p;  main(){  p = &x;  while (p++ != &y);  } | EVA finds that this program does not terminate. | CWE-232 |
| Undefined side-effects in expressions | int x, y, z, t, \*p, \*q; void main(int c) {  if (c&1)  y = x + x++;  p = c&2 ? &x : &t;  y = \*p + x++;  q = c&4 ? &z : &t;  y = \*q + x++; } | se.c:5:[kernel] warning: Unspecified sequence with side effect: /\* <- \*/ tmp = x; /\* x <- \*/ x ++; /\* y <- x tmp \*/ y = x + tmp;  se.c:5: ... undefined multiple accesses in expression. assert \separated(&x, &x); se.c:7: ... undefined multiple accesses in expression. assert \separated(&x, p); |  |
| Partially overlapping lvalue assignment | struct S { int a; int b; int c; }; struct T { int p; struct S s; }; union U { struct S s; struct T t; } u; void copy(struct S \*p, struct S \*q) {  \*p = \*q; } int main(int c, char \*\*v){  u.s.b = 1;  copy(&u.t.s, &u.s);  return u.t.s.a + u.t.s.b + u.t.s.c; } | partially overlapping lvalue assignment. assert p == q || \separated(p, q); | CWE-843 |
| Invalid function pointer access | extern void \* f1();  extern void f2(int);  void main(int c) {  int \* (\*p)(int);  void \*x, \*y;  p = &f1;  x = (\*p)(c);  p = &f2;  y = (\*p)(c);  } | valid\_function.c:9:[kernel] warning: pointer to function with incompatible type.assert \valid\_function(p);  valid\_function.c:9:[kernel] warning: Neither code nor specification for function f1, generating default assigns from the prototype  [value] using specification for function f1  valid\_function.c:12:[kernel] warning: pointer to function with incompatible type.assert \valid\_function(p);  valid\_function.c:12:[value] assertion 'Value,function\_pointer' got final status invalid. | CWE-908 |