```
========================================================================
Tutorial on Infer: A static program analyzer with some sample programs
========================================================================

-Prepared by: Sangharatna Godboley, School of Computing, NUS Singapore



------------------------------------------------------------------------------
1. To run dead_store.c program to find the errors related to deadstore
assisgnments and instructions.

=========================================================
 dead_store.c program With default command options
=========================================================


A. Program:

#include<stdio.h>
#include<assert.h>
int main()
{
int a,b = 0;
a =  __infer_nondet_int();
if (a > 5)
{
    b = 1;
}
else
{
    b = 2;
}
//Uncomment this line to
//fix the dead_store issue
//return b;
}

B. Run on buggy version

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run -o dead_store -- clang -c dead_store.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/dead_store


Analysis finished in 388mss

Found 2 issues

dead_store.c:9: error: DEAD_STORE
  The value written to &b (type int) is never used.
  7.   if (a > 5)
  8.   {
  9. >     b = 1;
  10.   }
  11.   else

dead_store.c:13: error: DEAD_STORE
  The value written to &b (type int) is never used.
  11.   else
  12.   {
  13. >     b = 2;
```

```
  14.   }
  15.   //Uncomment this line to


Summary of the reports

  DEAD_STORE: 2


C. Run on fixed version

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run -o dead_store -- clang -c dead_store.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/dead_store


Analysis finished in 360mss

  No issues found

--------------------------------------------------------------------------------


2. To run uninitialised_var.c program to find the errors where the varaibles
have not been initlised by any value.

=========================================================
 uninitialised_var program With default command options
=========================================================

A. Program

#include<stdio.h>
int main()
{
int  a;
/*Uncomment the below line
to fix the UNINITIALIZED_VALUE issue*/
//a =  __infer_nondet_int();
if (a > 5)
{
a = 1;
}
else
{
a = 2;
}
return a;
}

B. Run on buggy version

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run -o uninitialised_var -- clang -c uninitialised_var.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/uninitialised_var


Analysis finished in 384mss

Found 1 issue
```

```
uninitialised_var.c:8: error: UNINITIALIZED_VALUE
  The value read from a was never initialized.
  6.   to fix the UNINITIALIZED_VALUE issue*/
  7.   //a =  __infer_nondet_int();
  8. > if (a > 5)
  9.   {
  10.   a = 1;

Summary of the reports
UNINITIALIZED_VALUE: 1


C. Run on fixed version

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$ bin/infer run -o
uninitialised_var -- clang -c uninitialised_var.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/uninitialised_var


Analysis finished in 387mss

  No issues found
```

---

3. To run div_zero.c program to find the errors where the division operation’s denominator has value 0, in result the output will be undefined.

==========
div_zero.c program With command options
==========

A. Program

```
#include<stdio.h>
int main()
{
int  a, b = 0;
a = 8;
if (a > 5)
{

/*Uncomment the below line
to fix the div_by_zero issue
Please add the command option
"--enable-issue-type DIVIDE_BY_ZERO"*/

//b = 1;
a = a / b;
}
else
{
a = 2;
}
return a;
}
```

B. Run on buggy version

```
sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run --enable-issue-type DIVIDE_BY_ZERO -o div_zero -- clang -c
div_zero.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/div_zero


Analysis finished in 400mss

Found 1 issue

div_zero.c:15: error: DIVIDE_BY_ZERO
  Expression `b` could be zero at line 15, column 1.
  13.
  14.   //b = 1;
  15. > a = a / b;
  16.   }
  17.   else


Summary of the reports

  DIVIDE_BY_ZERO: 1


C. Run on fixed version

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$ bin/infer run --enable-
issue-type DIVIDE_BY_ZERO -o div_zero -- clang -c div_zero.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/div_zero


Analysis finished in 324mss

  No issues found
```

---

4. To run mixtype.c program to find the mix errors, with convenient command options.

```
==============================================
mixtype.c program runs
==============================================

A. Program

#include<stdio.h>
int main()
{
int  a, b = 0;
if (a > 5)
{
a = a / b;
}
else
{
b = 2;
}
return b;
}
```

B. Run with default command options where DEAD_STORE and UNINITIALIZED_VALUE types errors are enabled to report.

```
sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run -o mixtype -- clang -c mixtype.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/mixtype


Analysis finished in 338mss

Found 3 issues

mixtype.c:7: error: DEAD_STORE
  The value written to &a (type int) is never used.
  5.   if (a > 5)
  6.   {
  7. > a = a / b;
  8.   }
  9.   else

mixtype.c:5: error: UNINITIALIZED_VALUE
  The value read from a was never initialized.
  3.   {
  4.   int  a, b = 0;
  5. > if (a > 5)
  6.   {
  7.   a = a / b;

mixtype.c:7: error: UNINITIALIZED_VALUE
  The value read from a was never initialized.
  5.   if (a > 5)
  6.   {
  7. > a = a / b;
  8.   }
  9.   else


Summary of the reports

  UNINITIALIZED_VALUE: 2
           DEAD_STORE: 1
```

C. Run with convenient command options where DEAD_STORE and UNINITIALIZED_VALUE types errors are disabled and DIVIDE_BY_ZERO type error is enabled to report.

```
sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run --enable-issue-type DIVIDE_BY_ZERO
–disable-issue-type DEAD_STORE –disable-issue-type
 UNINITIALIZED_VALUE -o mixtype -- clang -c mixtype.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/mixtype


Analysis finished in 419mss

Found 1 issue

mixtype.c:7: error: DIVIDE_BY_ZERO
```

```
  Expression `b` could be zero at line 7, column 1.
  5.   if (a > 5)
  6.   {
  7. > a = a / b;
  8.   }
  9.   else


Summary of the reports

  DIVIDE_BY_ZERO: 1
sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
```

---

5. To run sample-teaching.c program to report the fale-positve errors. It means the errors are in infeasible path and Infer reports it as error, which ultimately is a boggus error.

A. Program

```
/* A sample program for
False Positve case*/
#include <stdio.h>
#include <klee/klee.h>
int main() {
    int x = 0;
    int y = 0;
    int a[1], b[5];
    b[0] = __infer_nondet_int();
    b[1] = __infer_nondet_int();
    b[2] = __infer_nondet_int();
    b[3] = __infer_nondet_int();
    b[4] = __infer_nondet_int();
    int i = 0;
    while(i < 5) {
       if (b[i] > 0)
       x = x + (i+2)*5;
       else x = x + 0;
       if (i == 4 && b[1] > 0)
       x = x + 0;
       else x = x + 30;
       i++;
    }
    int BOUND = 130;
    if (BOUND == x) {y++;}
    a[y] = 5;
    return 0;
}
```

B. Run the program with our convenient command with one more command option i.e. “--bufferoverrun” to find the such errors.

```
sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$
bin/infer run --bufferoverrun --enable-issue-type
DIVIDE_BY_ZERO --disable-issue-type DEAD_STORE
--disable-issue-type UNINITIALIZED_VALUE -o sample-teaching
 -- clang -c sample-teaching.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/sample-teaching


Analysis finished in 487mss
```

```
Found 1 issue

sample-teaching.c:26: error: BUFFER_OVERRUN_L2
  Offset: [0, 1] Size: 1.
  24.       int BOUND = 130;
  25.       if (BOUND == x) {y++;}
  26. >     a[y] = 5;
  27.       return 0;
  28.   }


Summary of the reports

  BUFFER_OVERRUN_L2: 1
-------------------------------------------------------------------------------

============================================================
Class Demo on zero_division.c program from  TOYOTA benchmark
============================================================

Command option is:
-------------------------------------------------------------------------------

bin/infer run --bufferoverrun --enable-issue-type DIVIDE_BY_ZERO --disable-
issue-type DEAD_STORE --disable-issue-type UNINITIALIZED_VALUE  -o zero_division
-- clang -c 01.w_Defects/zero_division.c
-------------------------------------------------------------------------------

Output:

sanghu@paella2018:~/Desktop/Infer/infer-linux64-v0.17.0$ bin/infer run
--bufferoverrun --enable-issue-type DIVIDE_BY_ZERO --disable-issue-type
DEAD_STORE --disable-issue-type UNINITIALIZED_VALUE  -o zero_division -- clang
-c 01.w_Defects/zero_division.c
Capturing in make/cc mode...
Found 1 source file to analyze in /home/sanghu/Desktop/Infer/infer-linux64-
v0.17.0/zero_division


Analysis finished in 744mss

Found 13 issues

01.w_Defects/zero_division.c:22: error: DIVIDE_BY_ZERO
  Expression `0` could be zero at line 22, column 2.
  20.       int dividend = 1000;
  21.       int ret;
  22. >     ret = dividend / 0;/*Tool should detect this line as error*/ /*
ERROR:division by zero */
  23.   }
  24.

01.w_Defects/zero_division.c:33: error: DIVIDE_BY_ZERO
  Expression `0` could be zero at line 33, column 2.
  31.       int dividend = 1000;
  32.       int ret;
  33. >     dividend /= 0;/*Tool should detect this line as error*/ /*
ERROR:division by zero */
  34.       ret = dividend;
  35.   }

01.w_Defects/zero_division.c:46: error: DIVIDE_BY_ZERO
  Expression `0` could be zero at line 46, column 2.
```

```
  44.       int dividend = 1000;
  45.       int ret;
  46. >     ret = dividend % 0;/*Tool should detect this line as error*/ /*
ERROR:division by zero */
  47.   }
  48.

01.w_Defects/zero_division.c:77: error: DIVIDE_BY_ZERO
  Expression `divisors[2]` could be zero at line 77, column 2.
  75.       int divisors[5] = {2, 1, 0, 3, 4};
  76.       int ret;
  77. >     ret = dividend / divisors[2];/*Tool should detect this line as
error*/ /* ERROR:division by zero */
  78.   }
  79.

01.w_Defects/zero_division.c:117: error: DIVIDE_BY_ZERO
  Expression `zero_division_007_s_gbl.divisor` could be zero at line 117, column
2.
  115.      int ret;
  116.      zero_division_007_func_001();
  117. >    ret = dividend / zero_division_007_s_gbl.divisor;/*Tool should
detect this line as error*/ /* ERROR:division by zero */
  118.   }
  119.

01.w_Defects/zero_division.c:128: error: DIVIDE_BY_ZERO
  Expression `0.` could be zero at line 128, column 2.
  126.      float dividend = 1000.0;
  127.      float ret;
  128. >    ret = dividend / 0.0;/*Tool should detect this line as error*/ /*
ERROR:division by zero */
  129.   }
  130.

01.w_Defects/zero_division.c:140: error: DIVIDE_BY_ZERO
  Expression `divisor` could be zero at line 140, column 2.
  138.      int divisor = 0;
  139.      int ret;
  140. >    ret = dividend / divisor;/*Tool should detect this line as
error*/ /* ERROR:division by zero */
  141.   }
  142.

01.w_Defects/zero_division.c:165: error: DIVIDE_BY_ZERO
  Expression `((2*divisor)-4)` could be zero at line 165, column 2.
  163.      int divisor = 2;
  164.      int ret;
  165. >    ret = dividend / (2 * divisor - 4);/*Tool should detect this line as
error*/ /* ERROR:division by zero */
  166.   }
  167.

01.w_Defects/zero_division.c:177: error: DIVIDE_BY_ZERO
  Expression `((divisor*divisor)-4)` could be zero at line 177, column 2.
  175.      int divisor = 2;
  176.      int ret;
  177. >    ret = dividend / (divisor * divisor - 4);/*Tool should detect this
line as error*/ /* ERROR:division by zero */
  178.
  179.   }

01.w_Defects/zero_division.c:194: error: DIVIDE_BY_ZERO
  Expression `returned by zero_division_013_func_001()` could be zero at line
```

```
194, column 2.
  192.      int dividend = 1000;
  193.      int ret;
  194. >    ret = dividend / zero_division_013_func_001();/*Tool should detect
this line as error*/ /* ERROR:division by zero */
  195.   }
  196.

...too many issues to display (limit=10 exceeded), please see
/home/sanghu/Desktop/Infer/infer-linux64-v0.17.0/zero_division/bugs.txt or run
`infer-explore` for the remaining issues.


Summary of the reports

    DIVIDE_BY_ZERO: 12
  NULL_DEREFERENCE: 1

--------------------------------------------------------------------------------

======================
Tutorial Assisgnments:
======================

Try Infer on programs from TOYOTA benchmark:

1. bit_shift.c
2. buffer_underrun_dynamic.c
3. func_pointer.c
4. invalid_memory_access.c
5. memory_leak.c
```