Overview of Red Hat Toolchain Projects Cauldron 2017

Martin Sebor

Senior Software Engineer, Red Hat

September 2017

Agenda

Red Hat Toolchain Team Projects

- Binutils
- GCC & libstdc++
- GDB
- Glibc
- Libabigail

Binutils/GCC – ELF Object Watermarking (Nick Clifton)

ELF Object Annotation

Records additional information in ELF object files for later static analysis.

Problem: A single program can consist of entities (object files or even functions) with mismatched requirements.

Use cases:

- Determine if all entities in a program use the same ABI (or don't use an aspect of it).
  - Type sizes. Arch-specific ABI variants. Doesn't use a certain type.
- Determine if all entities conform to a security policy.
  - Compile-time options. Link options/dependencies.
- Determine/verify run-time requirements.
  - Minimum hardware requirements. Minimum stack size.

Binutils/GCC – ELF Object Watermarking (cont'd)

GCC Plugin to Annotate ELF Objects

- Two sets of ELF Notes (`SHT_NOTE`) used to store information:
  - `.note.gnu.property.0`, `SHF_ALLOC`. Small allocatable section with data only used by `ld.so`.
  - `.gnu.build.attributes`, `SHF_GNU_BUILD_ATTRIBUTES` bit set. Non-allocatable section with additional information. Created by a GCC plugin.

More info: https://fedoraproject.org/wiki/Toolchain/Watermark
Sources: https://nickc.fedorapeople.org/annobin-2.1.tar.xz

GCC Projects

GCC Projects

- Stack Clash Mitigation (Jeff Law)
- Dynamic Range Evaluation (Aldy Hernandez, Andrew MacLeod)
- Diagnostic Infrastructure (David Malcolm)
- New And Improved GCC Warnings (Marek Polacek, Martin Sebor)
- Address Sanitizer (Jakub Jelinek)
- C++ 17 Core Language Support (Jason Merrill)
- Libstdc++ (Jonathan Wakely)

Stack Clash Mitigation (Jeff Law)

`-fstack-clash-protection`

- **Stack Clash:** Top of stack overwriting the bottom of the heap.
- New option to prevent stack clash attacks (beyond `-fstack-check`).
- Inserts probes every page to detect stack overflow into guard page.
- Fully supported on aarch*-*-*, powerpc*-*-*, s390*-*-*, i?86-*-*, and x86_64-*-*.
- `-param stack-clash-protection-guard-size=size` : Sets the size of the stack guard (default 4K).
- `-param stack-clash-protection-guard-interval=distance` : Sets the distance between stack probes (default 4K).
- **Status:** Under review, will be in GCC 8.0.

More info:

- CVE-2017-1000364
- https://gcc.gnu.org/ml/gcc-patches/2017-07/msg01971.html
- https://gcc.gnu.org/ml/gcc-patches/2017-07/msg01972.html
- https://gcc.gnu.org/ml/gcc-patches/2017-07/msg01974.html

Dynamic Range Evaluation (Aldy Hernandez, Andrew MacLeod)

Improvements to range information available throughout GCC.

- **Problem:** Range information available outside the VRP pass is limited and flow-insensitive, resulting in missed optimization opportunities and both false positive and false negative warnings in other passes that use ranges.
- **Goal:** Overcome limitations of the current VRP pass and representation and make range information fully general and more widely accessible and reliable outside it.
- Computes flow-sensitive range information on demand by traversing the CFG.
- Introduces new abstractions to easily manipulate (theoretically) unlimited sets of disjoint ranges.
- Replaces a single anti-range with multiple (2 - 3) ordinary ranges.
- Paves the way for decoupling the core of VRP from optimizations that make use of it.

Status: Work in progress, aiming for GCC 8.0.
Sources: `range-gen2` branch.

Diagnostic Infrastructure (David Malcolm)

Improvements to the quality of diagnostic messages.

- Function argument location enhancements.
- New fix-it hints for `-Wold-style-cast`.
- New fix-it suggestions for accessors for private/protected data members.
- New fix-it hints for missing C++ standard library `#include` directives.
- Better diagnostics for C++ template mismatches differing in subsets of arguments.

Diagnostic Infrastructure (David Malcolm)

Enhanced locations to be able to consistently underline function arguments.

```
warning: format ‘%s’ expects argument of type ‘char *’, but
argument 2 has type ‘int’ [-Wformat=]
   printf ("%s %i", 123, "foo");
            ~^      ~~~
             %d
```

Diagnostics (David Malcolm)

New fix-it hints for `-Wold-style-cast`.

```
warning: use of old-style cast to 'char' [-Wold-style-cast]
 char c = (char) i;
                 ^
          -------
           static_cast<char> (i)
```

Diagnostics (David Malcolm)

New fix-it suggestions for accessors for private/protected data members.

```
class C { int value; public: get_value () const { return value; }
};

error: ‘int C::value’ is private within this context
 int i = c.value;
           ^~~~~
note: declared private here
   int value;
       ^~~~~
note: field ‘int C::value’ can be accessed via ‘int C::get_value()
const’
 int i = c.value;
           ^~~~~
           get_value()
```

Diagnostics (David Malcolm)

New fix-it hints for missing C++ standard library #include directives.

```
int main ()
{
  std::cout << “Hello, World!\n”;
}
```

```
error: ‘cout’ is not a member of ‘std’
   std::cout << "Hello, World!\n";
         ^~~~
note: ‘std::cout’ is defined in header ‘<iostream>’; did you forget
to ‘#include <iostream>’?
+#include <iostream>
```

Diagnostics (David Malcolm)

Better diagnostics for C++ template mismatches differing in subsets of arguments.

- `-fdiagnostics-show-template-tree` : Print templates and their arguments in tree form.
- `-fno-elide-type`: Print all template arguments instead of eliding same ones.

Diagnostics (David Malcolm)

`-fdiagnostics-show-template-tree`

```
template <class T, class U, class V> struct A { };

void f (A<int, long, int>);

void g (A<long, int, int> a) {  f (a); }
```

```
error: could not convert ‘a’ from ‘A<long int,int,[...]>’ to
‘A<int,long int,[...]>’
  A<
     [long int != int],
     [int != long int],
     [...]>
 void g (A<long, int, int> a) { f (a); }
```

New And Improved GCC Warnings

Overview of Warning Options New Or Improved In GCC 8

- `-Wclass-memaccess` (Martin Sebor)
- `-Wduplicated-branches` (Marek Polacek)
- `-Wmultistatement-macros` (Marek Polacek)
- `-Wrestrict` (Martin Sebor)
- `-Wsizeof-pointer-div` (Marek Polacek)
- `-Wstringop-overflow` (Martin Sebor)
- `-Wstringop-truncation` (Martin Sebor)

`-Wclass-memaccess` (Martin Sebor)

Detects misuses of raw memory functions with C++ class objects.

Detects invalid uses of `memcpy`, `memmove`, and `memset` with objects of C++ class types:

- Reference corruption (overwriting reference data members).
- Vtbl pointer corruption (overwriting the virtual table pointer).
- Const-correctness violations (writing into const data members).
- Invariant violations (bypassing copy/move ctors and copy/move assignment).
- Encapsulation violations (writing into inaccessible members).

A superset of Clang `-Wdynamic-class-memaccess`.

Status: In GCC 8.0.

-Wclass-memaccess (Martin Sebor)

Example: Invariant and encapsulation violation.

```
struct S {
  …
  std::string str;
  …
};

void init (S *ps) { memset (ps, 0, sizeof *ps); }
```

```
warning: 'void* memset(void*, int, long unsigned int)' clearing an
object of type 'struct S' with no trivial copy-assignment; use
assignment or value-initialization instead [ -Wclass-memaccess]
```

-Wclass-memaccess (Martin Sebor)

Example: Using `realloc` with a non-trivially copyable type.

```
struct S {
  …
  S (const S&);
  …
};

void grow (S *ps, unsigned n) {  realloc (ps, n * sizeof *ps); }
```

```
warning: 'void* realloc(void*, long unsigned int)' moving an object
of non-trivially copyable type  'struct S'; use 'new' and 'delete'
instead [-Wclass-memaccess]
```

-Wduplicated-branches (Marek Polacek)

Detects branches of `if-else` statements with identical effects.

Satus: In GCC 8.0.

```
int g (int i)
{
  if (i == 0)
    return 1 + f ();
  else
    return f () + 1;
}
```

```
warning: this condition has identical branches [ -Wduplicated-branches]
```

-Wmultistatement-macros (Marek Polacek)

Detects unsafe macros that expand to two or more statements.

- **Problem:** Multi-statement macro invocations look like function calls but may not follow the same control flow in selection and iteration statements.
- Subset of CERT PRE10-C. Wrap multistatement macros in a do-while loop.
- **Status:** In GCC 8.

```
#define DEC(x, y) x++; y++;

int f (int x, int y)
{
  if (x)
    DEC (x, y);   // increments y unconditionally!
  …

warning: macro expands to multiple statements [ -Wmultistatement-macros]
```

`-Wrestrict` (Martin Sebor)

Detects overlapping copies in calls to library functions.

- **Problem:** `restrict` specifies that objects do not overlap. Compilers optimize code based on that guarantee (e.g., `memcpy`). Violations cause subtle bugs.
- GCC 7 detects a small subset of cases.
- GCC 8 enhances `-Wrestrict` to detect complex instances of overlaps.
- Handles most standard C library memory and string functions (e.g., `memcpy`, `strcpy`, and many others).
- **Status:** Under review.
- **To do:** Detect violations in calls to user-defined functions with `restrict`-qualified arguments.
- **Challenge:** How to determine non-trivial overlap in calls to user-defined functions with unknown semantics.

-Wrestrict (Martin Sebor)

Example: `strncpy` possible and certain partial overlap.

```
void f (char *d, unsigned p, unsigned n)
{
  strncpy (d, d + p, n);    // possible overlap

  if (n < 5 || 32 < n)      // n's range is [5, 32]
    n = 5;

  strncpy (d, d + 3, n);    // certain partial overlap
}
warning: 'strncpy' writing between 0 and 4294967295 bytes may overlap 1
or more bytes at offset 0 - 4294967295 [ -Wrestrict]

warning: 'strncpy' writing between 5 and 32 bytes overlaps 2 or more
bytes at offset 3 [ -Wrestrict]
```

`-Wsizeof-pointer-div` (Marek Polacek)

Detects suspicious uses of `sizeof` in array size computations.

- Warns about suspicious divisions of two `sizeof` expressions that divide the pointer size by the element size.
- Supplements `-Wsizeof-array-argument`.
- Implements CERT C ARR01-C. Do not apply the sizeof operator to a pointer when taking the size of an array.
- **Status:** In GCC 8.0.

-Wsizeof-pointer-div (Marek Polacek)

Example: Using `sizeof` with a pointer argument to compute number of elements.

```
unsigned sum (const int *a)
{
  unsigned n = 0;

  for (unsigned i = 0; i != sizeof a / sizeof a[0]; ++i)
    n += a[i];

  return n;
}
```

```
warning: division 'sizeof (const int *) / sizeof (int)' does not compute
the number of array elements [ -Wsizeof-pointer-div]
```

-Wsizeof-pointer-memaccess (Martin Sebor)

Enhanced to also detect likely mistakes involving `strncpy` and `strncat`.

Problem: Using the size of the source defeats the purpose of the bound. Code is silently accepted by GCC 7.
Status: Under review.

```
#define S "1234"

void h (char *d)
{
  strncpy (d, S, sizeof S);   // pointless use of strncpy
  …
```

```
warning: argument to ‘sizeof’ in ‘strncpy’ call is the same expression as
the source; did you mean to use the size of the destination?
[-Wsizeof-pointer-memaccess]
```

`-Wstringop-overflow` (Martin Sebor)

Enhanced to cover more cases of overflow and to also detect reading past the end.

- Similar to compiling with `-D_FORTIFY_SOURCE` but without runtime checks.
- GCC 7 detects buffer overflow in many library functions except `strncat`.
- GCC 8 alo detects instances of `strncat` overflow (superset of Clang's `-Wstrncat-size`).
- GCC 8 also detects reading past the end.
- **Status:** Parts in GCC 8.0, `strncat` enhancements under review.

-Wstringop-overflow (Martin Sebor)

Example showing diagnosing reading past the end.

```
#define foo_bar "foo\0bar"   // size is 8 bytes

int is_foo_bar (const char *s)
{
  return memcmp (s, foo_bar, sizeof foo_bar);
}

int f (void)
{
  return is_foo_bar ("foo");   // "foo" is only 4 bytes
}
```

```
warning: 'memcmp' reading 8 bytes from a region of size 4
[-Wstringop-overflow=]
```

`-Wstringop-truncation` (Martin Sebor)

Detects string truncation in calls to `strncpy`.

- **Problem:**
  - `strncpy` intended to completely fill buffers with data (struct `dirent::d_name`).
  - Does not guarantee nul-termination.
  - Commonly [mis]used as a bounded string copy.
- Option warns about calls to `strncpy` that result in truncation.
- Implements CERT STR03-C. Do not inadvertently truncate a string.
- **Status:** Under review.

-Wstringop-truncation (Martin Sebor)

Examples: using size of destination as the bound and `strlen` as the last argument to `strncpy`.

```
char a[32];

void h (void *d, const char *s)
{
  strncpy (a, s, sizeof a);      // no room for nul!
  …
  strncpy (d, s, strlen (s));    // ditto
  …
```

```
warning: ‘strncpy’ specified bound 32 equals destination size
[-Wstringop-truncation]
```

```
warning: ‘strncpy’ output truncated before terminating nul copying as
many bytes from a string as its length [ -Wstringop-truncation]
```

Address Sanitizer (Jakub Jelinek)

`-fsanitize=pointer-overflow`

- Detects overflow in pointer arithmetic expression at runtime.
- Not the same thing as out-of-bounds pointer detection!
- Provides a quick “sanity” check for pointer expressions.
- Common use case: Performing arithmetic on null pointers.
- **Caveat:** Pointer offsets are treated as signed even if they are unsigned in the source so adding an offset >= `SSIZE_MAX` subtracts.

Address Sanitizer (Jakub Jelinek)

Example: Null iterator overflow.

```
iterator insert (iterator pos, size_t n, const T &x)
{
  iterator end = this->end ();
  if (pos == end) {
    this->append (n, x);
    return end - 1;   // Overflow when end is null
  }
  …
}
```

Other GCC Improvements (Martin Sebor)

Attributes `read_only`, `read_write`, `write_only`, and `no_side_effect`

Tell GCC how user-defined functions access objects referenced by arguments.
`no_side_effect` is just like `pure` but allows write accesses to objects referenced by pointers.
(Might `noescape` instead be better?)

Goals:

- Open up optimization opportunities similar to built-in functions.
- Detect misuses of user-defined functions:
  - out-of-bounds accesses (`-Wstringop-overflow`)
  - uninitialized reads (`-Wuninitialized`)
  - unused but set variables (`-Wunused-by-set-variable`)
- Enable checking of Variable Length Array arguments.

Attributes `read_only`, `write_only`, etc. (Martin Sebor)

Example of hoisting invariant member function calls out of loops.

Declare `std::string::find()` like so:

```
std::string::size_type
std::string::find (const std::string&) const noexcept
  __attribute__ ((no_side_effect, read_only (0), read_only (1)));

void foo (const std::string &a)
{
  const std::string b = read_string ();
  for (int i = 0; i != n; ++i) {
    size_t pos = a.find (b);    // see call hoisted out of the loop
    …
```

Attributes `read_only`, `write_only`, etc. (Martin Sebor)

Example of out-of-bounds access detection.

```
ssize_t my_read (void*, size_t) __attribute__ (( write_only (1, 2) ));

void foo (void)
{
  char buf[80];
  ssize_t nr = my_read (buf, 1024);
  …
```

```
warning: ‘my_read’ writing 1024 bytes into a region of size 80 overflows
the destination [-Wstringop-overflow=]
```

C++ 17 Core Language Support (Jason Merrill)

As an example: Overhaul of support for lambdas in templates.

Improves the internal representation of C++ lambdas to better handle their interaction with templates.

```
template <class... T>
auto f (auto i) {
  return ([i]{ return T (i); }() + ...);
}

int main () {
  // compute the value of: 1234 + (unsigned char)1234
  printf ("%i\n", f<int, unsigned char>(1234));   // prints 1444
}
```

Libstdc++ (Jonathan Wakely)

The GNU C++ Standard Library

- Adds AddressSanitizer annotations to `std::vector`.
- Adds deduction guides (P0433R2, partial)
- Supports `std::experimental::source_location` (N4519).
- Many bug fixes.

GDB Projects

- GCC C++ Compile And Execute (Alexandre Oliva, Keith Seitz, Phil Muldoon)
- Statement Frontier Notes and Location Views (Alexandre Oliva)

GCC/GDB – C++ Compile And Execute

(Alex Oliva, Keith Seitz, Phil Muldoon)

Compiling and injecting C++ code in GDB

- GCC: available in GCC 7 (originally on branch: `origin/aoliva/libcp1`)
- GDB: available on branch: `origin/users/pmuldoon/c++compile`
- Works with existing `compile` command:
  - `compile code` *`source-code`*
  - `compile file` *`file-name`*
  - `compile print` *`expression`*

More info: C++ Support In libcc1

GCC/GDB – Statement Frontier Notes and Location Views (Alexandre Oliva)

Improving debugging optimized code.

First published at GCC Summit 2010.
Finally implemented in GCC 8.0.
Relies on GCC and Gas enhancements.

Statement Frontier Notes populate `is_stmt` column of line number tables, pointing to code statements.
Each `is_stmt` value associated with source location tied to variable binding events.
Location views provide a way for mapping variable assignments to source locations even if code is rearranged.

To learn more:

- Attend Alex's talk on *Consistent Views at Recommended Breakpoints*
- Read Statement Frontier Notes and Location Views

Glibc Projects

- Glibc 2.26 Released (August 2017)
- Glibc 2.25 Released (February 2017)
- Glibc Malloc improvements (DJ Delorie)
- Glibc `/etc/resolv.conf` reloading (Florian Weimer)

Malloc Improvements (DJ Delorie)

`malloc` per-thread cache

- Adds a per-thread cache for a 17% improvement.
- Adds a `malloc` capture/replay simulator to benchmark hard-to-reproduce application behaviors via archived workloads.

More info:

- Glibc Malloc Internals
- `malloc` per-thread cache: benchmarks

Glibc `/etc/resolv.conf` reloading (Florian Weimer)

Reloads changed configuration.

- Glibc DNS stub now automatically reloads `/etc/resolv.conf` to pick up changed configuration settings.
- Finally provides a solution to a long-standing request/bug (dating back 17 years).

More info: Glibc bug 984.

Libabigail (Dodji Seketelli)

Verifying ABI and API compatibility

- Libabigail compares binaries and finds ABI-impacting changes (e.g., data or function type or size changes, new or removed data members, etc.)
- Report includes source file location, function, data, and type names, offsets and sizes.
- Linux kernel and kernel module ABI comparison (`kabidiff`).
- Two kernel trees compared in about 30 seconds and 3GB RAM.

More info: https://sourceware.org/libabigail
Kabidiff: https://sourceware.org/git/gitweb.cgi?p=libabigail.git;h=refs/heads/dodji/kabidiff

Questions?

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos