Resolving the almost decade old checker dependency issue in the Clang Static Analyzer

Kristóf Umann
dkszelethus@gmail.com
Eötvös Loránd University, Budapest
Ericsson Hungary
April 8., 2019
The original problem: easy-to-mess-up command line interface

```bash
clang -cc1 -analyze myfile.cpp \
    -analyzer-checker=cplusplus.InnerPointer \
    -analyzer-config note-as-warning=true
...meant to be notes-as-warnings
```

```bash
clang -cc1 -analyze myfile.cpp \
    -analyzer-checker=cplusplus.InnerPointer \
    -analyzer-config unix.Malloc:Optimist=true
...meant to be Optimistic
```
No warnings, no errors, the analyzer simply doesn’t do what you intended...
Bug unearthed: "The Checker Naming Bug"

- Multiple checker objects could receive the same name
- Incorrect checker names in bug reports
- Errors while parsing checker configurations
Real-life problems coming from the Checker Naming Bug

```
clang -cc1 -analyze myfile.cpp \
  -analyzer-checker=cplusplus.InnerPointer \
  -analyzer-config unix.Malloc:Optimistic=true
```
Real-life problems coming from the Checker Naming Bug

```
clang -cc1 -analyze myfile.cpp \
  -analyzer-checker=cplusplus.InnerPointer \
  -analyzer-config cplusplus.InnerPointer:Optimistic=true
```
InnerPointerChecker and MallocChecker have the same name!

- Turns out InnerPointerChecker depends on MallocChecker!
- InnerPointerChecker enables both itself and MallocChecker
- Fixing this bug implies the need to reimplement dependencies...
clang -cc1 -analyze myfile.cpp \
-analyzer-checker=cplusplus.InnerPointer \
-analyzer-config unix.Malloc:Optimistic=true
clang -cc1 -analyze myfile.cpp \
-analyzer-checker=cplusplus.InnerPointer \
-analyzer-config cplusplus.InnerPointer:Optimistic=true
How do we solve this?

CString Modeling object?

- unix.Malloc
- unix.cstring.BadSizeArg
- alpha.unix.cstring.OutOfBounds
- unix.cstring.NullArg
- alpha.unix.cstring.NotNullTerminated
- alpha.unix.cstring.BufferOverlap

- cplusplus.NewDelete
- unix.MismatchedDeallocator
- cplusplus.InnerPointer

- c

--
Be able to represent dependencies with a directed tree
Resolve dependencies at a higher level

- Declare dependencies in TableGen
- Don’t allow checkers to enable more than one checker
- Make sure dependencies are enabled in the correct order

def InnerPointerChecker : Checker<"InnerPointer">, HelpText<"Looks for pointers to temp. strings">, Dependencies<[DynamicMemoryModeling]>, Documentation<NotDocumented>;}
Conclusion

- We are able to list checker dependencies
- We can now list and verify checker options
- Checker names won’t depend on how we invoke the analyzer
- Plugins can now depend on builtin checkers
- Already in trunk!
Thank you for your attention!