Technical Specification

Clang Randstruct Project

Author:	Tim Pugh
Date:	2018-11-30
Version:	1.0

Document Control

Document location

Location
PDX.EDU Domain->G-Suite->Clang-Drive (Administrator: Tim Pugh)

Author

Position	Name	Contact no
Team Lead	Tim Pugh	503-739-3231

Revision history

Version	Issue date	Author/editor	Description/Summary of changes
1.0	2018-11-30	Tim Pugh	Initial Specification

Reviewed by

Version	Issue date	Name	Position	Review date

Approvals

Version	Issue date	Name	Position	Approval date
1.0	2018-11-30	Tim Pugh	Team Lead	2018-12-02

Related documents

Document	Location
All Files	https://drive.google.com/drive/u/0/folders/0AFmz2KEALoKcUk9PVA

Table of Contents

1 INTRODUCTION

1.1 Objectives

Clang is a compiler front end for the C, C++, Objective-C languages whose original author and team lead was Chris Lattner. Clang has been heavily contributed to by companies such as Apple Inc, Google, and many individual developers around the world as an open source project.

We've been tasked by Kees Cook to produce a re-implementation of the GCC compiler plugin randstruct which randomizes the layout of C structs for Clang.

Below are the objectives of the re-implementation:

Develop full randomization. All structures marked with `__randomize_layout` have field positions randomized, including bit fields.

Develop a 'performance-sensitive' mode. Best-effort limited randomization to cache-line (64-byte) size region, keeping adjacent bit fields together.

Develop an automatic structure selection method (e.g. all functions pointers). Disabled with `__no_randomize_layout`.

Develop regression tests.

Randomization seed needs to be externally created or known before building.

Committed into LLVM and Clang.

1.2 Scope

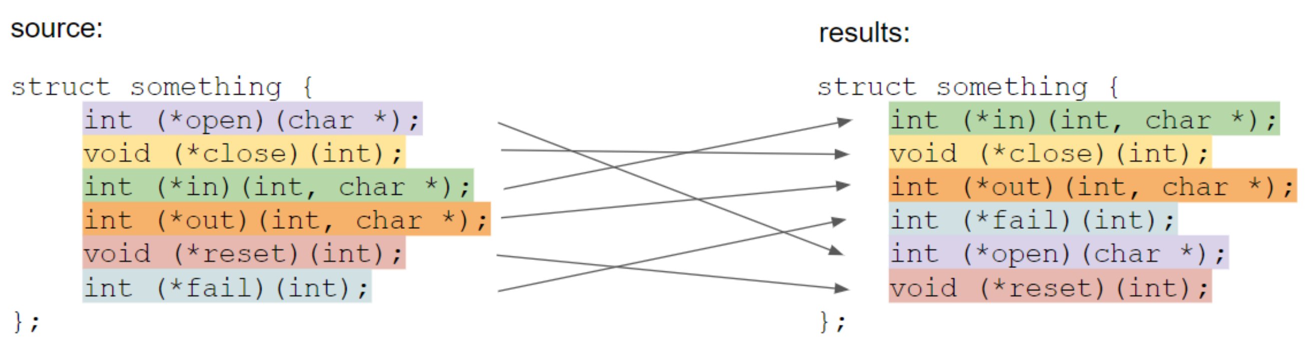
The scope of the project entails a team of 7 software engineers (Connor Kuehl, Jeff Takahashi, Jordan Cantrell, Nikk Forbus, James Foster, Cole Nixon and Tim Pugh) from Portland State dedicating each 12+ hours per week researching the feasibility of implementing the project objects, implementing the feasible objectives, presenting our findings to our sponsor and submitting our code for commitment into the Clang project. The work is scheduled in 2 phases:

Oct. 29th , 2018 – Dec. 3rd , 2018: Research, infrastructure, prototyping and mid project presentation.
Jan. 8th, 2019 – Mar. 18th , 2019: Research, development, verification validation, submission upstream, and final presentation

2 Functionality

2.1 Functionality

Currently Clang does not have a mechanism to randomize C structures. Additional overhead for the inclusion of the randomization process will be incurred but are expected to be kept minimal. The mechanism for randomizing the struct/class/union fields will have a compilation system in place to turn the feature on or off depending on what the developer using Clang to compile their code so desires. The functionality of the system will re-organize the layout of a struct/class/union when compiled and not of the source code. Below is a representation of the functionality:



source:

```
struct something {
    int (*open)(char *);
    void (*close)(int);
    int (*in)(int, char *);
    int (*out)(int, char *);
    void (*reset)(int);
    int (*fail)(int);
};
```

results:

```
struct something {
    int (*in)(int, char *);
    void (*close)(int);
    int (*out)(int, char *);
    int (*fail)(int);
    int (*open)(char *);
    void (*reset)(int);
};
```

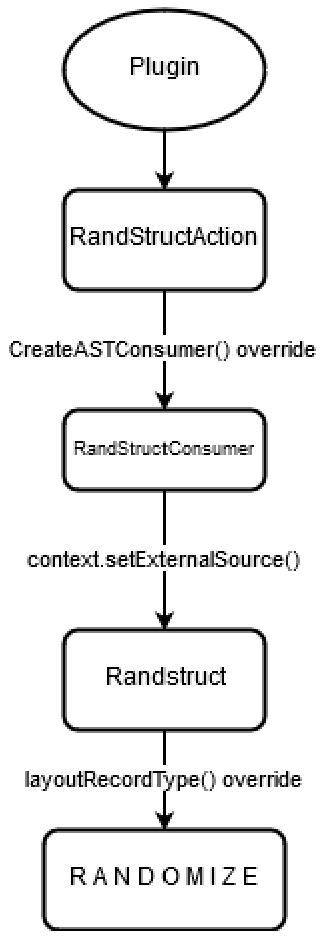
3 Achitecture

3.1 Architecture Overview

`Randstruct` is an `ExternalASTSource` that overrides the `LayoutRecordType()` callback. It's within this callback that we must *perform the entire construction (alignment, padding, everything) of the record layout while randomizing its fields.*

4 Design

4.1 Object Diagram


Plugin
RandStructAction
CreateASTConsumer() override
RandStructConsumer
context.setExternalSource()
Randstruct
layoutRecordType() override
RANDOMIZE

4.2 Other Components

5 CODE

5.1 File Names and Structure

The executable file names will remain the same as upstream clang as we are contributing to a pre-existing project and are choosing not to re-name anything that may interfere with the future compilation of the project. We will adapt our program to work with the current file names and structure within Clang.

In the event we name files, we will use “randstruct” or have it conformed to current standards.

5.2 Executable File Names and Structure

The executable file names will remain the same as upstream clang as we are contributing to a pre-existing project and are choosing not to re-name anything that may interfere with the future compilation of the project. We will adapt our program to work with the current file names and structure within Clang.

In the event we name files, we will use "randstruct" or have it conformed to current standards.