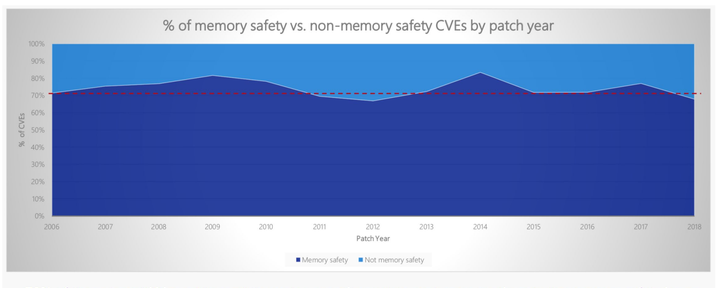
Automated Conversion of Legacy Code to Checked C

Aravind Machiry*, Hasan Touma+, Ray Chen+, Michael Hicks+
*University of California, Santa Barbara. machiry@cs.ucsb.edu
+University of Maryland and Correct Computation, Inc.
{htouma@,{rchen,mwh}@cs}.umd.edu

It is almost 2020, but we still have memory corruption vulnerabilities.



~70% of the vulnerabilities addressed through a security update each year continue to be memory safety issues

Lets use safe languages

Safe by design: Prevents memory corruption vulnerabilities.

Rust Programming Language

Golang

What about **Legacy code**? Not feasible to rewrite.

Can we retroactively make the legacy code safe?

Address Sanitizer (ASan)

Slow (>= 50%).

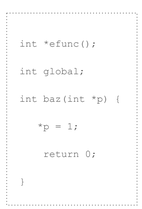
Not backward compatible and need runtime changes.

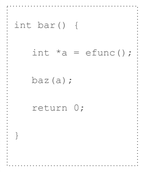
Checked C

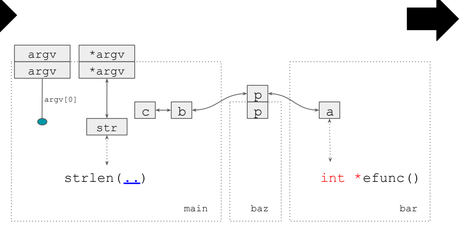
Fast

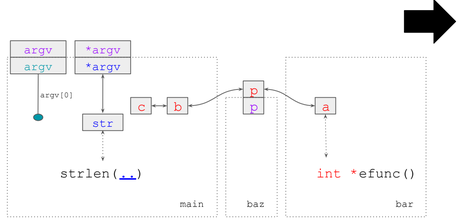
Backward compatible.

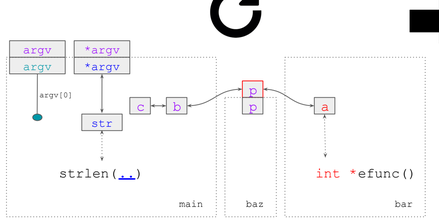
_Ptr
_Array_ptr
_Nt_array_ptr

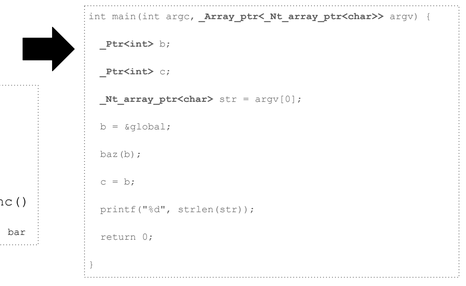












Input source files

1.Constraint Graph Creation

2. Constraint Solving

3. Iterative Constraint Refinement

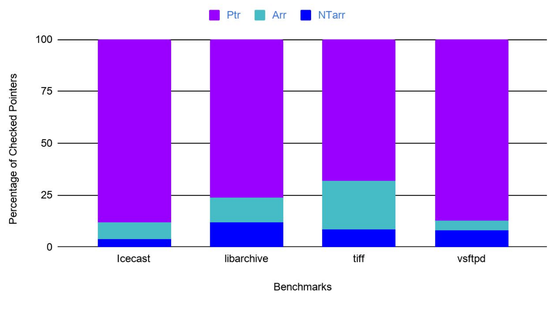
4. Source files with Checked C annotations

Overview of our approach to automatically convert legacy code to Checked C

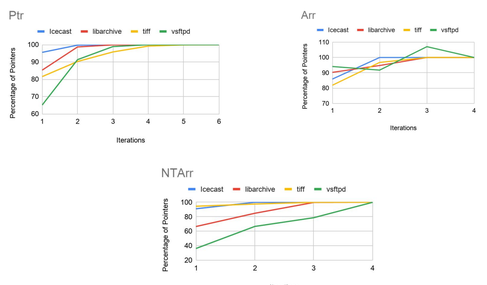
Evaluation

Name	Description	SLOC (KLOC)	Total Pointers	Checked Pointers
Icecast	Media server	18	3,218	1,769 (54.97%)
libarchive	Compression Library	151	14,637	11,266 (76.97%)
tiff	Image Utilities	72	7,120	5,007 (70.32%)
vsftpd	FTP Server	16	2,035	1,789 (87.91%)
Total		257	27,010	19,831 (73.42%)

Detection Rate



Iterative refinement effectiveness



Conclusions and Future work

- Automated conversion of legacy code to Checked C is feasible and preliminary results are encouraging.
- Work being done on inferring bounds for array variables and checked regions.
- Available online: https://github.com/microsoft/checkedc-clang/tree/master/tools/checked-c-convert