Softbound: <http://dl.acm.org/citation.cfm?id=1542504> (only sequential programs)

CETS: <http://www.cis.upenn.edu/acg/papers/ismm10_cets.pdf> (Only sequential programs - our offloading technique solves the problem of multithreading races to a particular location)

CCured: <http://dl.acm.org/citation.cfm?id=1065892> (Fat pointer representation, source level analysis but restricted to sequential programs. Possibly we get better performance)

CIRM: <http://pgas11.rice.edu/papers/PirkelbauerEtAl-UPC-Error-Detect-PGAS11.pdf> (Our paper is a follow-up to this one.)

Butterfly Analysis: <http://www.cs.cmu.edu/~mgoodste/research/butterfly_asplos10.pdf> (dataflow analysis, not memory checking)

MemSafe: <http://www.ece.umd.edu/~simpsom/papers/scam2010.pdf> (only sequential programs)

Bounded Model-Checking: <http://users.ecs.soton.ac.uk/lcc08r/esbmc/papers/ifm2012.pdf>

CheckPointer: <http://140.239.210.162/Company/Publications/CheckPointer.pdf> (program has to be data race free, otherwise this won’t work)

WIT: <http://dl.acm.org/citation.cfm?id=1398074&CFID=104871057&CFTOKEN=88595254> (Changes memory layout, conservative static analysis)

MSCC: <http://www.seclab.cs.sunysb.edu/seclab/pubs/fse04.pdf> (Doesn’t detect sub-object overflows, and arbitrary casts – according to Softbound)

HAQu: <http://people.engr.ncsu.edu/jtuck/papers/lee_haqu_hpca_2011.pdf> (Only handles mudflap which is object-based, not pointer-based scheme – coverage problem)

Fail Safe C: <http://dl.acm.org/citation.cfm?id=1542505> (Fat Pointers, Memory layout compatibility problems)

Purify: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.184.8081

Valgrind: <http://cs.ucsb.edu/~arch/spr07-seminar/papers/valgrind2007.pdf> (parallel threads, but only one thread runs at a time)

<http://valgrind.org/docs/manual/drd-manual.html> -- tool in valgrind for supporting multithreaded programs – only supports POSIX threading primitives

Dynamic Binary Instrumentation schemes typically can’t detect out-of-bounds within objects (??) because of lack of high-level information. Slow.

MemCheck: (doesn’t detect dangling pointers to reallocated mem locations – the technique used to track temporal errors is not as good as the lock-key impl)

TaintCheck:

Helgrind: Multithreaded tool in Valgrind: <http://valgrind.org/docs/manual/hg-manual.html>

DRD: Debugging multithreaded programs: <http://valgrind.org/docs/manual/drd-manual.html>

DieHard: <http://dl.acm.org/citation.cfm?id=1134000> - Doesn’t handle stack corruption. Only does heap errors.

Runtime Verification of C Memory Safety: <http://dl.acm.org/citation.cfm?id=1690881>

Backwards-compatible bounds checking: <http://wwwhomes.doc.ic.ac.uk/~phjk/Publications/BoundsCheckingForC.pdf> (Coverage problems -- handling of function params is not exactly proper)

Low-overhead memory leak detection using statistic prof: <http://dl.acm.org/citation.cfm?id=1024412> (only detects memory leaks, but interesting sampling technique)

Efficient detection of all pointer and array access errors: <http://dl.acm.org/citation.cfm?id=178446&CFID=104871057&CFTOKEN=88595254> (Fat pointer representation. No talk of multi-threaded programs)

Static detection of dynamic memory errors: <http://www.cs.virginia.edu/~evans/pubs/pldi96.pdf> (annotates interfaces to

detect errors… something along those lines)

Cyclone: A safe dialect of C: <http://static.usenix.org/event/usenix02/jim.html> - restrictions on pointer arithmetic. Fat pointers

Hardware Debuggers:

iWatcher: Efﬁcient Architectural Support for Software Debugging: <http://dl.acm.org/citation.cfm?id=1061269>

AccMon: Automatically detecting memory related bugs via Program-Counter based Invariants: <http://dl.acm.org/citation.cfm?id=1038950>

Intel Thread Checker:

DIDUCE

Eraser

SafeMem

DISE

ABCD – Array bound check – on Java. http://dl.acm.org/citation.cfm?id=349342&CFID=104871057&CFTOKEN=88595254

Hardbound: <http://dl.acm.org/citation.cfm?id=1346295&CFID=104871057&CFTOKEN=88595254>

Memory safety without runtime checks or garbage collection – no runtime checks, only static analysis of type-safe C programs.

Secure execution via Program Shepherding: <http://groups.csail.mit.edu/cag/rio/security-usenix.pdf>

HeapMon: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5388734&tag=1>

(Only monitors heap accesses, requires hardware support in terms of cache bits)

Low-cost, concurrent checking of pointer and array accesses in C programs: <http://www.cs.cmu.edu/afs/cs/Web/People/lba/LBA_reading_group/papers/low-cost-concurrent-checking_97.pdf>

Softbound Refs on ACM:

<http://dl.acm.org/citation.cfm?id=1542504>

Also check multithreaded error checking programs