Reliable Return Address Stack: Microarchitectural Features to Guard Against Stack Smashing

Dong Ye, Micha Moffie and David Kaeli
ECE Department
Northeastern University

dye, mmoffie, kaeli@ece.neu.edu

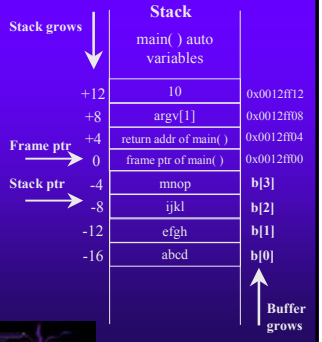
Outline

- Introduction to stack smashing
- Return Address Stack (RAS)
- Reliable Return Address Stack (RRAS)
- Performance evaluation
- Conclusions and future work

Program Stack of a C Process

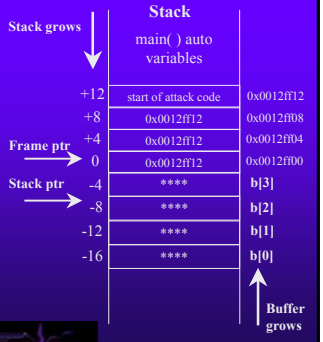
```
main(int argc, char **argv)
{
   …
   foo(argv[1], 10);
   …
}

void foo(int i, char *s) {
   char b[16];
   strcpy(b, s);
   ……
}
```



Stack Smashing Scheme

1. Overwrite an unchecked buffer
2. Change the return address, making it point to the attack code
3. Attack code will be executed upon return



Return Address Stack (RAS)

- A microarchitectural unit that predicts function return addresses (performs jump prediction)
  - Very accurate prediction
    - Original version: >90% correct
    - With some repair mechanisms: 99% correct
  - Captures the common case of function calls
    - Last-In-First-Out (LIFO)
  - Works well and is widely deployed

RAS – Enforce Security?

Can we use RAS to provide the correct return address (RA)?

- Cannot be overwritten
  - invisible to viruses
- But there is no guarantee of correctness
- Incorrect predictions due to:
  - Finite size of RAS entries
  - Multithreading
  - Speculative execution
  - Non-LIFO function calls

Solution: Reliable Return Address Stack (RRAS)

- Reserved RRAS spill area in memory
- Per-thread stack context
- Built-in fully associative logic and additional storage to find the correct RA on the entire RAS upon each function return
  - Matches function entry address if it is non-recursive
  - Matches return address if call is recursive or if the function entry address is not available yet

RRAS Design

Upon call:

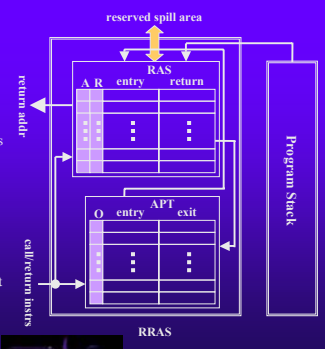
- Check for recursion
- Check for direct recursion
- Push onto RAS

APT table:

- Record entry/exit address pairs of called functions

Upon return:

- Check for recursion and first call with exit address
- Load RA from program stack if necessary
- Locate RA on RAS and send it to the return instr.
- Update APT and RAS if necessary



Performance Tuning

- Recursion demands additional effort to guarantee correctness
  - "A" tag on the RAS and "O" tag on the APT are used to differentiate them from one another
- A direct recursion can flood the RAS and incur spill/restore operations
  - "R" tag on RAS identifies direct recursion and reuses existing RAS record

Performance Evaluation

- Overhead incurred only upon call/return
  - Instruction mix analysis of SPEC CPU 2000

benchmark	% of call and return
bzip2	1.70%
crafty	2.03%
eon	4.09%
gcc	0.41%
gzip	0.99%
mcf	2.93%
parser	3.71%
perlbmk	2.18%
twolf	1.57%
vortex	3.62%
vpr	1.47%
wupwise	1.16%
art	0.05%
fma3d	0.74%
apsi	0.15%

Conclusions and Future Work

- RRAS can guarantee the correctness of return addresses
- Further evaluation
  - Performance
  - Formal proof
- Guard against smashing function pointers

Northeastern University
Computer Architecture Research Group

This work is supported by NSF award CISE CSA0310891