Paper Reading Group

Summer 2017

Dresden, 6th April 2016

Administrativia

- Every Thursday, 11:10 AM, APB 3105
- Presentation & discussion of one paper
- Staff papers voted on
- Mailing List (see website)
- Usually: Pizza

Rules

- Pick one paper
    - *Explore* field of research by following related work
    - *Present* research field in 75 minutes talk
    - *Write* 8 page survey paper
- Pick own topic and write a paper suitable for *workshop submission*

Rules — Johns Hopkins University

- ~~Pick one paper~~
  - ~~*Explore* field of research by following related work~~
  - ~~*Present* research field in 75 minutes talk~~
  - ~~*Write* 8 page survey paper~~
- ~~Pick own topic and write a paper suitable for *workshop submission*~~

Rules — TUD

- One paper presentation per student
- Pick a paper related to systems research (suggestions on the website)
- Prepare 15 – 30 minute presentation
  - In English
  - Show that you understood the paper
  - Extra knowledge (related work) may be helpful
  - Prepare questions/issues for discussion
- Choose a few papers you would like to present until next week

Rules — TUD (continued)

For papers you do not present:

- Write a short paper summary (10 - 15 sentences)
  - Explain what you understood
  - Raise questions/issues
  - Mention things you liked/disliked
- Use your own words, do *not* just copy from the paper!
- Questions are the basis for discussion
- Send to `prg-summary@os.inf.tu-dresden.de` as *plain text*
- Deadline: day before the meeting (Wednesday noon)
- Say if you want pizza in your mail (defaults to no!)

Example Mail

```
I want pizza this week
Summary:
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.  Ut purus elit, vestibulum ut,
placerat ac, adipiscing vitae, felis.  Curabitur dictum gravida mauris.  Nam arcu libero,
nonummy eget, consectetuer id, vulputate a, magna.  Donec vehicula augue eu neque.  Pellentesque
habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.  Mauris ut leo.
Cras viverra metus rhoncus sem.  Nulla et lectus vestibulum urna fringilla ultrices.  Phasellus
eu tellus sit amet tortor gravida placerat.  Integer sapien est, iaculis in, pretium quis,
viverra ac, nunc.  Praesent eget sem vel leo ultrices bibendum.  Aenean faucibus.  Morbi dolor
nulla, malesuada eu, pulvinar at, mollis ac, nulla.  Curabitur auctor semper nulla.  Donec
varius orci eget risus.  Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam.  Duis
eget orci sit amet orci dignissim rutrum.
Questions / Discussion Points:

  - Why mollis ac, nulla semper?
  - Dolor et less than egestas?
  - I really liked eget sem vel leo!
```

Sample Summaries

Summary 1

The Paper introduces Generic Object Protection (GOP), a software-based fault tolerance mechanism. The basic idea of this approach is to supplement critical data structures with error-detecting codes (EDC).

To avoid manual augmentation of existing programs with such EDCs the authors employ aspect-oriented programming using AspectC++. Thereby the compiler can insert checks/updates of the EDCs before accessing/after modifying a protected object.

Whole-program analysis (i.e. compile- and link-time optimization) is used reduce the mechanism's overhead. According to the evaluation said overhead amounts to about 20% in code size, 1% in memory usage, and 0.01% in total runtime when using GOP on an undisclosed subset of kernel classes in the L4/Fiasco.OC microkernel. In those experiments 60% of the unprotected kernel's failures are avoided by GOP.

Summary 1 (Ctd)

Pros/Likes

- relevant topic
- code examples
- actual implementation of the proposed scheme
- substantiate used fault model with reference to large-scale study

Cons/Dislikes

- no proper introduction to AspectC++
- useless/unexplained information (table 1, most of figure 2)
- evaluation
  - too little information about setup of case study (section 5)
  - no full exploration/sampling of possible faults in evaluation ([. . .]we [. . .] randomly inject one bit flip in each run)
  - no actual runtimes given

Summary 1 (Ctd.)

Questions

- Section 5 states 3. GOP: [. . .] we applied the GOP to protect all data members of 23 kernel classes.. 23 classes out of how many? How were those 23 chosen anyway? Same question about the used applications.
- In section 4 the 'lid' is said to be A local identifier describing the order of each function call. What is the order of a function call?

Summary (2)

The paper describes how one can use an aspect oriented extension of C++, called AspectC++ to harden OS kernel against transient hardware faults.
The paper advocates for the SW-based FT, because they enable a possibility for a more fine grained control over the parts of the system, which are protected on a software level. Potentially, this may result in a better performance, even in comparison to HW based solutions.
State of the art software based techniques are not suitable to protect the operating system kernel, since most of them see the kernel as RCB. The paper addresses this shortcoming.
The authors use AspectC++ to extend the most critical OS data structures with Hamming encoding. It is up to the programmer to decide which particular kernel objects to protect. The methods to annotate the source code are typical for AOP.

Summary 2 (Ctd.)

I liked an idea of "Whole-Program Analysis". This is an optimisation which allows to avoid some redundant checks of the protected state and may significantly reduce the overhead from the hardening.
A particular thing, which I did not understand is following statement: "the real execution time ... is each three orders of magnitude higher than their pure instruction counts."
Additionally, I'm not happy with the way how the authors present the overhead from the kernel hardening. They say that the overhead asserted in the abstract includes mostly time spent in a user level code. Such evaluation makes little sense, because if you do not let the hardened code run, you can't see slowdown from it.
Also, I have doubts regarding the applicability of the whole approach, because it seems that it improves robustness of the kernel "somewhat", but it is hard to quantify the increased level of reliability.