OS Reliability

Lecture #03:	Monday, 12 February 2007
Lecturer:	Emmet Witchel
Scribe:	Jungho Jo
Reviewer:	Min Kyu Jeong

1 Mondrix: Memory Isolation for Linux using Mondriaan Memory Protection

1.1 Introduction

This paper deals with a version of Linux kernel with Mondriaan Memory Protection(MMP). MMP provides fine-grained protection with backwards compatibility for operating system, ISAs and program models with an additional hardware support. The paper discusses the design and an evaluation of MMP.

MMP provides memory protection that enables robust OS operation in case of error or malicious attack. MMP overlays an address space with multiple disjoint protection domains, each with a unique set of permissions. Every thread is associated with exactly one protection domain, and any number of threads can be in the same protection domain at the same time. The domains are protected by hardware support by using protection look-aside buffer(PLB) that stores protection information.

Cross-domain calling in MMP needs two-way guarantee. A thread can enter a callee's domain only at switch gate, and a thread can only return at the instruction following the call to the corresponding switch gate. Domain protection is changed at the first instruction of callee function, which is a switch gate. Since the caller do not have a callee's domain information, we do not have to speculatively change the domain protection until we execute the switch gate.

Mondrix memory supervisor is the most privileged software layer that runs under an existing kernel. The supervisor consists of two parts, a top and a bottom. The top layer

[*]Copyright 2007 Jungho Jo and Mattan Erez, all rights reserved. This work may be reproduced and redistributed, in whole or in part, without prior written permission, provided all copies cite the original source of the document including the names of the copyright holders and "The University of Texas at Austin EE382V Spring 2007, Computer Architecture: User System Interplay".

writes the permissions tables in memory. The bottom layer does the rest of the jobs, including presenting a memory protection interface to the kernel.

To provide robustness to OS, the Mondrix partitions the Linux kernel. Each modules is loaded into arbitrary domain using MMP. If the collection of functions are considered as module, then the kernel developers can allow that collection to control the memory permission.

The design of MMP is evaluated in two ways; functional evaluation and performance evaluation. To evaluate functionality, the memory corruption in kernel is emulated by fault injection experiments. The performance was evaluated by using SimICS and Bochs machine simulators.

1.2 Reference

Witchel, E.; Rhee, J.; Asanovic, K, "Mondrix:Memory Isolation for Linux using Mondriaan Memory Protection", Proceedings of the twentieth ACM symposium on Operating systems principles, 2005

2 Discussion Points

2.1 What is the problem being solved?

- The authors try to achieve OS kernel robustness & fault tolerance by following techniques:
  - Implementing fine-grained memory protection
  - Combining hardware & software to achieve memory isolation
- Comment by Dr. Witchel

  We should note that fine-grained memory protection by combining HW/SW is a mechanism to achieve OS robustness.

  The goal of the paper is to achieve OS robustness and fault tolerance. The robustness does not targeted malicious behaviors, whereas fault tolerance targeted wide range of threat model.

  Usually software is known more vulnerable to attack. For example, plug-ins are code or data added to existing program for more functionality. Plug-ins are loaded into address space and OS creates symbolic link to active them. The advantage of using plug-ins is they have good performance and they use same programming model as main program. However, plug-ins does not guarantee any kind of protection to the system. One bad write to the memory can cause a crash to the whole system whether the behavior of the plug-in is malicious or not.

To avoid crash due to bad memory write, message passing type kernel can provide good protection. However, such OS suffers performance loss. We still want to good performance and compatibility of legacy code. This is ongoing issue of kernel design.

2.2 Who are the intended users?

- OS developers & users.
- Microarchitects

2.3 What is unique about the suggested solution?

- Mondriaan Memory Protection
  - Memory protection mechanism is provided by hardware support.
  - Switch/return gates for cross-domain calling. There are many issues regarding setting a privilege level in cross-domain calling. One of the important issue is that we do not have information whether `'call'` instruction is cross-domain call or not until run-time. It is important especially in system functions or driver code that usually reside in different domain than user program.
  - Stack should be protected in different way. Protection domain is not sufficient for stack because different thread in same domain can access same stack.
- Memory Supervisor
- Compartmentalizing Linux

2.4 How is the idea evaluated?

- The authors injected faults to the kernel code that possibly crashes the system. They showed that MMP can detect the faults in the kernel code. They also found out that the kernel code has some bugs in managing the memory.
- The authors measured the performance by simulation.
  - They used two system emulators - SimICS & Bochs
    - The simulators have some bugs.
    - The two simulation results are cross checked.
  - They run simulations to evaluate the performance of MMP in quantitatively & qualitatively.
  - They evaluated hardware & software overhead by using MMP.

2.5 Was the evaluation in line with the user requirements?

- Hardware overhead (area/power) is small enough to justify adding hardware support for MMP
- Managed code comparison

2.6 Was technology a factor in the problem or solution?

- No, technology has little to do with the topic.

2.7 Were new tools or software technique was introduced?

- Entire kernel was modified to support MMP.
- Memory supervisor that runs under the kernel was introduced.

2.8 How may users with other requirements be affected?

- We can think about how this idea can interact with virtualization. If MMP and virtualization is applied together, which layer should we apply MMP? Should we apply MMP hardware & software to real machine or should we apply it to virtual layer?
- The multiple processor users should consider MMP and coherency of memory together.

2.9 How can the discussion of this paper be generalized in the context of the class?

- Mondrix provides backward compatibility of programming model. The code written in unsafe language can still be used.
- This paper deals with robustness from the malicious attack, which is in line with current trend.
- The MMP explores combination of hardware & software.
- The paper did not introduced recovery mechanism which is an important issue.
- Hardware can provide more fine-grained fault detection than software, but we have to consider its overhead.