Mapping Unikernels with TAG based architectures

Akilan Selvacoumar

Mathematics and Computer Sciences
Heriot Watt University

Year 1 progression report of:
Doctor of Philosophy

October 2022

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Akilan Selvacoumar
October 2022

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Chapter 2

Motivation

Chapter 3

Research Questions

The following section talks about research questions:

- Which areas can Unikernels provide performance gains for TAG based architectures in comparison to the same implementation built using a monolithic OS?
- Does using Enclaves inside Unikernels provide a isolation mechanism within Unikernels, maintain lightweightness characterestics, and what would be the performance differnence between using Intel SGX and a open source implementation such as Timber V?
- Due to lesser depencies in Unikernels does that mean lesser TAG policies are required for the appication ?
- Can Unikernel provide sufficient performance in such a way that a dedicated processor is not required for processing TAGS ?
- Does Unikernels with TAGS provide a secure and elastic environment ?

]

Chapter 4

Literature Review

The literature review is split into 3 sections. The first section talks about the papers surveyed for Unikernels and the 2nd section talks about papers surveyed for TAG based architectures and the third sections talks about the possible incentives of combining them both which helps answer the research questions stated (TODO: Add reference to research question section).

4.1 TAG based architecture survey

The following was a survey conducted on exisisting TAG based implementations and the recent survey based on TAG based architectures [2] published in 2022 was a good staring point to understand about various implementations of TAG based archtectures with the high level metrits and limitations. The following section provides our own version of the Survey to help decide the best implementations to answer the research questions (//TODO reference research questions chapter).

Before deep diving into TAG based architecture implementations it is important to answer what is a TAG based architecture ? and the high level of various categories of various TAG based architectures.

Tagged architectures are a prominent class of hardware security primitives that augment data and code words with tags. The tags, which function as the security metadata about memory, are created before the program is loaded. Then, at runtime, the hardware enforces security policies on the tags to provide safety guarantees. The advantage being tags automate the secure and efficient management of security metadata.

Tags policies as designed to address mostly:

- Type and memory corruption
- Integer overflows

- Thread safety
- Buffer overflows

TAG policies can be categorised into 5 main categories which is:

- Information-low control (IFC) policies
- Dynamic information-low tracking (DIFT) policies
- Capability models
- Programmable architectures

According to the TAG based architecture survey [2] there are 37 published efforts on TAG based architectures over the past decade and 20 published efforts preceding that.

4.1.1 Timder V [9]

It is a tagged memory architecture for flexible and efficient isolation of code and data on small embedded systems. The TAG isolation is augmented with a memory protection unit to isolate induvidual processes. Timber V is compatible with exsisting code. The contributions of the paper are:

- Efficient tagged memory architecture for isolated execution on low-end processors.
- Concept introducted called stack interleaving that allows efficient and dynamic memory management.
- Lightweight shared memory between enclaves.
- Efficient shared MPU (i.e Memory Protection Unit) design.

4.1.2 ARM MTE [1]

The ARMv8.5-Memory Tagging Extension (MTE) aims to increase the memory safety written for unsafe languages without requiring source code changes and in certain cases without recompilation. It generally foicuses on the bounds checking use case, Though it provides limited tags which means it can only provide probablilistic overflow detection. It is one of the latest commercial incarnations of memory-safety-focused tagged architectures.

4.1.3 D-RI5CY [5]

It provides a design a design and implementation of a hardware dynamic information flow tracking (DIFT) architecture for RISC-V processor cores. The paper presents a low overhead implementation of DIFT that is specialized for low-end embedded systems for IOT applications. The following are high level contributions:

- Design f D-RI5CY, A DIFT-protected implementation of the RI5CY processor core. The paper implements the modification of the DIFT TAG propogation and TAG checking mechanism in a way that is transparent to the execution of the regular instructions.
- Concept introducted called stack interleaving that allows efficient and dynamic memory management.
- Lightweight shared memory between enclaves.
- Efficient shared MPU (i.e Memory Protection Unit) design.

4.1.4 HyperFlow [3]

It is a design and security implementation that offers security assurance because it is implemented using a security-typed hardware description language. It allows complex information flow policies to be configured at run time. The paper introduces ChiselFlow, a new secure hardware description language. The contribution of the paper includes:

- Processor architecture and implementation designed for timing-safe information flow security.
- Complere RISC-V intruction set extended with intructions for information flow control.
- Verified at design time with a hardware description language.
- Novel representations of lattices that can be implemented in hardware efficiently.

Hyperlow implements a nonmalleable IFC policy using tags. To eliminate timing side channels, the processor tracks the tag of the currently executing code and lushes caches, TLB, branch predictor, and other microarchitectural state on changes in the conidentiality or integrity tag of the running code. The modifications to avoid timing side channels seem more extensive than those to add tags. The authors report overheads in cycles per instruction of between 1% and 69%, largely due to padding the multiply operation to the worst-case number of cycles.

4.1.5 SDMP [6]

This paper focuses on designing metadata tag based stack-protection security policies for general purpose tagged architecture. The policies specifically exploit the natural locality of dynamic program call graphs to achieve cacheability of the metadata rules that they require. The simple Return Address Protection policy has a performance overhead of 1.2% but just protects return addresses. The two richer policies present, Static Authorities and Depth Isolation, provide object-level protection for all stack objects. When enforcing memory safety, The Static Authorities policy has a performance overhead of 5.7% and the Depth Isolation policy has a performance overhead of 4.5%. The contribution of the paper includes:

- The formulation of a range of stack protection policies within the SDMP model.
- Three optimizations for the stack policies: Lazy Tagging, Lazy Clearing and Cache Line Tagging.
- The performance modeling results of the policies on a standard benchmark set, including the impact of the proposed optimizations.

4.1.6 Typed Architecture [4]

This paper introduces Typed Architectures, a high-efficiency, low-cost execution sub- strate for dynamic scripting languages, where each data variable retains high-level type information at an ISA level. Typed Architectures calculate and check the dynamic type of each variable implicitly in hardware, rather than explicitly in software. Typed Architectures provide hardware support for flexible yet efficient type tag extraction and insertion, capturing common data layout patterns of tag- value pairs. The evaluation using a fully synthesizable RISC-V RTL design on FPGA shows that Typed Architectures achieve mean speedups of 11.2% and 9.9% with minimum speedups of 32.6% and 43.5% for two production- grade scripting engines for JavaScript and Lua. The contribution of the paper includes:

- ISA extension to efficiently manage type tags in hardware, which can be flexibly applied to multiple scripting languages and engines.
- Design and implement the Typed Architecture pipeline, which effectively reduces the overhead of dynamic type checking at low hardware cost.
- Prototype the proposed processor architecture using a fully synthesizable RTL model to execute two production-grade scripting engines with large inputs on FPGA (executing over 274 billion instructions in total) and provide a more accurate estimate of area and power using a TSMC 40nm standard cell library.

4.1.7 Dover [7]

It is a secure processor that extends the conventional CPU with a Policy EXecution co-processor (PEX). PEX maintains metadata of every word assesible by the application processor. PEX enforces software-defined policies at the granularity of each instruction executed by the AP(i.e application process) CPU. Hardware interlocks enforce strict separation between code and data for user-land and policy-related. The Dover system has a dover specialized kernel and modifications to the GCC toolchain which can implement a wide range security and safety policies on top exsisting C based applications.

4.1.8 CHERI [8]

CHERI (Capability Hardware Enhanced RISC Instructions) extends conventional processor Instruction-Set Architectures (ISAs) with architectural capabilities to enable fine-grained memory protection and highly scalable software compartmentalization. CHERI is a hybrid capability architecture that can combine capabilities with conventional MMU(i.e Memory Management Unit) based systems. The contribution of the following project include:

- ISA changes to introduce architecture capabilities.
- New microarchitecture proving that capabilities can be implemented efficiently in hardware. Support for efficient tagged memory to protect capabilities and compress capabilities to reduce memory overhead.
- Newly designed software construction model for that uses capability to provide fine grain memory protection and scalable software compartmentalization.
- Langauge and Compiler extension to use capabilities for C and C++.
- OS extensions to use (and support application use of) fine-grained memory protection (spatial, referential, and (non-stack) temporal memory safety) and abstraction extensions to support scalable software compartmentalization.

Chapter 5

Expirements

Chapter 6

Research Goals

Chapter 7

Research Timeline

Chapter 8

Conclusion

References

[1] (2019). 1 Armv8.5-A Memory Tagging Extension. https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Arm_Memory_Tagging_Extension_Whitepaper.pdf?revision=ef3521b9-322c-4536-a800-5ee35a0e7665&la=en&hash=D510ED84099D3B8AA34723AC110D48E3A28FA8D6. [Accessed 20-Oct-2022].

[2] (2022). TAG: Tagged Architecture Guide | ACM Computing Surveys — dl.acm.org. https://dl.acm.org/doi/abs/10.1145/3533704. [Accessed 20-Oct-2022].

[3] Ferraiuolo, A., Zhao, M., Myers, A. C., and Suh, G. E. (2018). Hyperflow: A processor architecture for nonmalleable, timing-safe information flow security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1583–1600, New York, NY, USA. Association for Computing Machinery.

[4] Kim, C., Kim, J., Kim, S., Kim, D., Kim, N., Na, G., Oh, Y. H., Cho, H. G., and Lee, J. W. (2017). Typed architectures: Architectural support for lightweight scripting. *SIGARCH Comput. Archit. News*, 45(1):77–90.

[5] Palmiero, C., Di Guglielmo, G., Lavagno, L., and Carloni, L. P. (2018). Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–7. ISSN: 2377-6943.

[6] Roessler, N. and DeHon, A. (2018). Protecting the stack with metadata policies and tagged hardware. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 478–495.

[7] Sullivan, G. T., DeHon, A., Milburn, S., Boling, E., Ciaffi, M., Rosenberg, J., and Sutherland, A. (2017). The dover inherently secure processor. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–5.

[8] Watson, R. N., Woodruff, J., Neumann, P. G., Moore, S. W., Anderson, J., Chisnall, D., Dave, N., Davis, B., Gudka, K., Laurie, B., Murdoch, S. J., Norton, R., Roe, M., Son, S., and Vadera, M. (2015). Cheri: A hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy*, pages 20–37.

[9] Weiser, S., Werner, M., Brasser, F., Malenko, M., Mangard, S., and Sadeghi, A.-R. (2019). TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA. Internet Society.

References

[1] (2019). 1 Armv8.5-A Memory Tagging Extension. https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Arm_Memory_Tagging_Extension_Whitepaper.pdf?revision=ef3521b9-322c-4536-a800-5ee35a0e7665&la=en&hash=D510ED84099D3B8AA34723AC110D48E3A28FA8D6. [Accessed 20-Oct-2022].

[2] (2022). TAG: Tagged Architecture Guide | ACM Computing Surveys — dl.acm.org. https://dl.acm.org/doi/abs/10.1145/3533704. [Accessed 20-Oct-2022].

[3] Ferraiuolo, A., Zhao, M., Myers, A. C., and Suh, G. E. (2018). Hyperflow: A processor architecture for nonmalleable, timing-safe information flow security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1583–1600, New York, NY, USA. Association for Computing Machinery.

[4] Kim, C., Kim, J., Kim, S., Kim, D., Kim, N., Na, G., Oh, Y. H., Cho, H. G., and Lee, J. W. (2017). Typed architectures: Architectural support for lightweight scripting. *SIGARCH Comput. Archit. News*, 45(1):77–90.

[5] Palmiero, C., Di Guglielmo, G., Lavagno, L., and Carloni, L. P. (2018). Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pages 1–7. ISSN: 2377-6943.

[6] Roessler, N. and DeHon, A. (2018). Protecting the stack with metadata policies and tagged hardware. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 478–495.

[7] Sullivan, G. T., DeHon, A., Milburn, S., Boling, E., Ciaffi, M., Rosenberg, J., and Sutherland, A. (2017). The dover inherently secure processor. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–5.

[8] Watson, R. N., Woodruff, J., Neumann, P. G., Moore, S. W., Anderson, J., Chisnall, D., Dave, N., Davis, B., Gudka, K., Laurie, B., Murdoch, S. J., Norton, R., Roe, M., Son, S., and Vadera, M. (2015). Cheri: A hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy*, pages 20–37.

[9] Weiser, S., Werner, M., Brasser, F., Malenko, M., Mangard, S., and Sadeghi, A.-R. (2019). TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA. Internet Society.

Index