The Reduced Address Space (RAS) for Application Memory Authentication

David Champagne, Reouven Elbaz and Ruby B. Lee

Princeton University, USA

Introduction

Background:

- TPM, XOM, AEGIS, SP, SecureBlue want to provide trust in an application's computations and protect private information.
- An adversary corrupting the memory space of an application can affect the trustworthiness of its computations.

Security Model:

- Threats:
 - Physical attacks: Tampering with bus data or memory chip
 - Software (SW) Attacks: Compromised OS
- Assumptions:
 - Processor chip is the security perimeter
 - Application to protect is correctly written (no SW vulnerabilities)
 - On-chip engine can authenticate initial state of application

Objective

Provide application memory authentication: What the application reads from a memory location is what it last wrote there.

Outline

Introduction to memory integrity trees

Past Work:

- Building a tree over the physical address space (PAS Tree)
 Insecure: branch splicing attack
- Building a tree over the virtual address space (VAS Tree)
 Impractical

Proposed Approach

- A novel Reduced Address Space (RAS)
- ✓ Building a tree over the RAS (RAS Tree)
- Managing the RAS Tree with the Tree Management Unit (TMU)
- Performance evaluation

Conclusion

Addressing Nodes in an Integrity Tree



Outline

Introduction to memory integrity trees

Past Work:

Building a tree over the physical address space (PAS Tree)
 Insecure: branch splicing attack

Building a tree over the virtual address space (VAS Tree)
 Impractical

Proposed Approach

- A novel Reduced Address Space (RAS)
- ✓ Building a tree over the RAS (RAS Tree)
- Managing the RAS Tree with the Tree Management Unit (TMU)
- Performance evaluation

Conclusion

Physical Address Space (PAS) Tree

- Majority of past work implements a PAS Tree, where nodes form a contiguous memory region in the PAS
- Problem: with untrusted OS, branch splicing attack can be carried out

<u>Branch splicing attack</u>: substitution of a leaf data block for another via page table corruption

On-Chip Root Recomputation











Branch Splicing Attack on PAS Tree



Root
$$\stackrel{\textbf{?}}{=} branch_{\boldsymbol{\phi}}(D\boldsymbol{\phi})$$



Outline

Introduction to memory integrity trees

Past Work:

Building a tree over the physical address space (PAS Tree)
 Insecure: branch splicing attack

Building a tree over the virtual address space (VAS Tree)
 Impractical

Proposed Approach

- A novel Reduced Address Space (RAS)
- ✓ Building a tree over the RAS (RAS Tree)
- Managing the RAS Tree with the Tree Management Unit (TMU)
- Performance evaluation

Conclusion

Virtual Address Space (VAS) Tree is Too Wide

- When OS is untrusted, VAS trees are implemented:
 - Tree nodes form a contiguous memory region in the VAS
- Problem: Tree must cover huge segment of memory space



This leads to huge integrity trees:

= Unused memory region

- Very large memory capacity overhead
- Very large initialization latencies

Tree span	32-bit	48-bit	64-bit
Total memory footprint	Gigabytes	Terabytes	Exabytes!
Initialization latency	Minutes	Months	Millenia!

Branch Splicing Fails on VAS Tree



Outline

Introduction to memory integrity trees

Past Work:

- Building a tree over the physical address space (PAS Tree)
 Insecure: branch splicing attack
- Building a tree over the virtual address space (VAS Tree)
 Impractical

Proposed Approach

- A novel Reduced Address Space (RAS)
- ✓ Building a tree over the RAS (RAS Tree)
- Managing the RAS Tree with the Tree Management Unit (TMU)
- Performance evaluation

Conclusion

Proposed Approach

- The Reduced Address Space (RAS): a novel address space containing only pages necessary to the application's execution
 - RAS expands dynamically to fit the application's memory needs
 - RAS contains compact descriptions of memory regions not mapped in RAS, the Unmapped Page Ranges (UPRs)
- Compute integrity tree over RAS (a RAS tree) for dramatic reduction of memory and initialization overheads
- When application touches a previously unused page, on-chip logic expands RAS and adds branch to RAS tree

The Reduced Address Space (RAS)

RAS initially contains the application image authenticated at load-time.

- The RAS index maps a virtual page into the Reduced Address Space
- Tree is built over RAS so it spans only useful pages
- Page mapped into RAS when application touches it for the first time.
- Tree root expanded to span new page



Tree Expansion



Branch Splicing Fails on RAS Tree



Wes: REJECT D61 ass D11

Attack Detected

Tree Management Unit



RAS addr = RAS_index || offset

Cost Evaluation

Low memory overhead (3 to 10 orders of magnitude reduction over 64-bit VAS!)



Reductions in initialization latency proportional to memory overhead reduction

Performance hit: 5% vs. no integrity tree, 2.5% vs. VAS tree, < 2.5% vs. PAS tree</p>

Conclusions

- We described the branch splicing attack on memory integrity trees.
- We provided practical application memory authentication resistant to branch splicing despite an untrusted OS
- We reduced memory capacity overhead by 3 to 10 orders of magnitude
- We reduced initialization latency by 3 to 10 orders of magnitude



Questions?

References

[D. Champagne, R. Elbaz et al.] "The Reduced Address Space (RAS) for Application Memory Authentication" In Proceedings of the 11th Information Security Conference (ISC'08), September 2008.

[R. Elbaz, D. Champagne et al.] "TEC-Tree: A Low Cost and Parallelizable Tree for Efficient Defense against Memory Replay Attacks," Cryptographic Hardware and embedded systems (CHES), September 2007.

[B. Gassend et al.] "Caches and Merkle Trees for Efficient Memory Authentication," High Performance Computer Architecture (HPCA-9), February 2003.

[R. Merkle] "Protocols for Public Key Cryptosystems," IEEE Symposium on Security and Privacy, 1980.

[G. E. Suh et al.] "AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing," Proc. of the 17th Int'l Conf. on Supercomputing (ICS), 2003.

[C. Yan et al.] "Improving Cost, Performance, and Security of Memory Encryption and Authentication", Int'l Symposium on Computer Architecture (ISCA'06), June 2006.

BACKUP SLIDES

Root Recomputation Equations

Leaf Position to Verify	Root Recomputation Performed by On-chip Authentication Engine
1	$COMP1 = h\{ h[h(D1 D2) H4] H2 \}$
2	$COMP2 = h\{ h[h(D1 D2) H4] H2 \}$
3	$COMP3 = h\{ h[H3 h(D3 D4)] H2 \}$
4	$COMP4 = h\{ h[H3 h(D3 D4)] H2 \}$
5	$\underline{COMP5} = h\{ H1 h[h(D5 D6) H6] \}$
6	$COMP6 = h\{ H1 h[h(D5 D6) H6] \}$
7	$COMP7 = h\{ H1 h[H5 h(D7 D8)] \}$
8	$\underline{COMP8} = h\{ H1 \parallel h[H5 \parallel h(D7 \parallel D8)] \}$







leaf to verify (fetched from memory)

COMPi = root re**COMP**utation equation for leaf position i

Branch Splicing Attack





 Normal verification flow
 Verification flow under attack from malicious OS

XXX Data controlled by malicious OS

P(X) = physical address of X V(X) = virtual address of X