Sego: Pervasive Trusted Metadata for Efficiently Verified Untrusted System Services

Youngjin Kwon, Alan Dunn, Michael Lee, Owen Hofmann, Yuanzhong Xu, Emmett Witchel
Securing OS is difficult

OS vulnerabilities in 2014 from national vulnerability database (NVD)

- Mac OS
- iOS
- Linux Kernel
- Window 8.1
- Windows Server 2012

- Large attack surfaces
- System calls
- ioctl interface
- 3rd party device driver
Securing OS is not enough

Vulnerability distribution in 2014 from NVD

- Getting root leads to control OS
- Privilege escalation vulnerability
- Many APPs run with root permission
Protecting application from malicious OS

• With trusted hypervisor
  Overshadow (ASPLOS 2008)
  TrustVisor (IEEE S&P 2010)
  InkTag (ASPLOS 2013)
  **Sego (ASPLOS 2016)**

• With compiler instrumentation
  VirtualGhost (ASPLOS 2014)

• With hardware (SGX) support
  Haven (OSDI 2014)
Outline

• Previous system

• Sego eliminates encryption and hashing

• Sego provides crash consistency and recovery

• Conclusion
How do previous systems work?
Trust model

System overview

- APP
- Guest operating system
- Hypervisor
- Hardware
- Secure APP
- Sego library

- Interpose syscall
- cooperate with hypervisor

Sego library

hypercall

Trusted

Untrusted
Hypervisor encrypts memory for secrecy

Hypervisor

APP

OS

Hypervisor

Plaintext

Ciphertext

Software

RAM

storage
Hypervisor encrypts memory for secrecy

1. APP reads/writes memory page

APP reads/writes memory page

1. APP reads/writes memory page

1. APP reads/writes memory page

1. APP reads/writes memory page

1. APP reads/writes memory page

1. APP reads/writes memory page

1. APP reads/writes memory page
Hypervisor encrypts memory for secrecy

1. APP reads/writes memory page
2. OS wants to swap page
Hypervisor encrypts memory for secrecy

1. APP reads/writes memory page
2. OS wants to swap page
3. Hypervisor blocks OS
   a) Encrypts page
Hypervisor encrypts memory for secrecy

1. APP reads/writes memory page
2. OS wants to swap page
3. Hypervisor blocks OS
   a) Encrypts page
4. OS swaps encrypted page
Hypervisor encrypts memory for secrecy

1. APP reads/writes memory page
2. OS wants to swap page
3. Hypervisor blocks OS
   a) Encrypts page
4. OS swaps encrypted page
Hypervisor hashes memory for integrity
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

---

Diagram:
- Hypervisor
- RAM
- Storage
- APP
- OS
- Metadata
- Plaintext
- Ciphertext
- Hash
- Software
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata
2. OS wants to swap page
3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page

4. OS swaps the encrypted page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
    a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
    a) Encrypts page
    b) Hashes page

4. OS swaps the encrypted page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page

4. OS swaps the encrypted page

5. APP accesses page
   a) OS swaps in
   b) HYP checks hash
   c) HYP decrypts page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page

4. OS swaps the encrypted page

5. APP accesses page
   a) OS swaps in
   b) HYP checks hash
   c) HYP decrypts page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page

4. OS swaps the encrypted page

5. APP accesses page
   a) OS swaps in
   b) HYP checks hash
   c) HYP decrypts page
Hypervisor hashes memory for integrity

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS wants to swap page

3. Hypervisor blocks OS
   a) Encrypts page
   b) Hashes page

4. OS swaps the encrypted page

5. APP accesses page
   a) OS swaps in
   b) HYP checks hash
   c) HYP decrypts page
Performance cost of encryption and hashing

- Performance of encryption and hashing
  - AES-NI (GCM) supported in processor
  - 800MB/s - 1.2 GB/s
- Performance of a single IO device
  - Commodity SSD : 520MB/s
  - Fusion-io ioDrive : 1GB ~ 1.5GB/s
- IO bandwidth can overwhelm encryption bandwidth!
OS Memory Services

- Modern services require OS to touch memory
  - Transparent page sharing
    - Multiple virtual machines consume less memory
    - Overshadow/InkTag cannot support it
  - Memory compaction
    - OS defragments memory for large pages
    - Better TLB utilization
- We must make OS access to APP pages more efficient
Sego eliminates encryption and hashing by using trusted metadata
Replace encryption and hashing with hypercalls
Replace encryption and hashing with hypercalls

1. APP reads/writes memory page
   a) HYP maintains metadata
Replace encryption and hashing with hypercalls

1. APP reads/writes memory page
   a) HYP maintains metadata

- APP
- OS
- Sego hypervisor

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>mA</td>
<td>mB</td>
<td>mC</td>
</tr>
</tbody>
</table>

protected data
Software

Hypervisor memory
Replace encryption and hashing with hypercalls

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS is not allowed to access protected memory pages
Replace encryption and hashing with hypercalls

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS is not allowed to access protected memory pages

3. OS sends hypercall to move memory pages
Replace encryption and hashing with hypercalls

1. APP reads/writes memory page
   a) HYP maintains metadata

2. OS is not allowed to access protected memory pages

3. OS sends hypercall to move memory pages

4. Hypervisor moves the memory page
Sego persists data with metadata

- Virtualized block device
  - Virtual hard disk/SSD
  - Sees/controls all I/O
  - Buffers guest IO in host memory

- Hypervisor storage
  - Invisible to OS
  - Holds trusted metadata
Sego persists data with metadata

- Virtualized block device
- Virtual hard disk/SSD
- Sees/controls all I/O
- Buffers guest IO in host memory

- Hypervisor storage
  - Invisible to OS
  - Holds trusted metadata

- APP
- OS
- Sego hypervisor
- Virtualized block device
- IO buffer
- Hypervisor memory
- Hypervisor storage
- OS storage
- Software
- protected data
Sego persists data with metadata

- Virtualized block device
- Virtual hard disk/SSD
- Sees/controls all I/O
- Buffers guest IO in host memory
- Hypervisor storage
  - Invisible to OS
  - Holds trusted metadata
Pervasive trusted metadata

- Metadata is everywhere
  - To protect data in memory: hypervisor memory
  - To protect data in storage: hypervisor storage

- Metadata is shared
  - Hypervisor and virtualized block device share metadata
Sego protects data with pervasive metadata

- Metadata in memory: for Hypervisor protecting data
- Metadata in storage: for virtualized block device protecting data
Sequential read

- InkTag/Overshadow
  - Protect app by encryption and hashing

- SSD (250MB/s)
  - 13 ~ 15% improvement by removing encryption and hashing

- Hard disk
  - IO batching optimization

![Graph showing sequential read performance for Linux-VM, InkTag, and Sego on Hard disk and SSD.](image)
OS touches protected memory

**Transparent page sharing**
scan 100 pages at every 20 milliseconds

**Memory compaction**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Linux-VM</th>
<th>InkTag</th>
<th>Sego</th>
</tr>
</thead>
<tbody>
<tr>
<td>429.mcf</td>
<td>1</td>
<td>2.1</td>
<td>1.05</td>
</tr>
<tr>
<td>470.lbm</td>
<td>1</td>
<td>1.61</td>
<td>1.05</td>
</tr>
<tr>
<td>graph analysis</td>
<td>1</td>
<td>1.39</td>
<td>1.01</td>
</tr>
<tr>
<td>Micro benchmark (Sequential read)</td>
<td>1</td>
<td>1.25</td>
<td>1.06</td>
</tr>
</tbody>
</table>
Sego provides crash consistency and secure recovery without trusting OS

**Guest OS crash**

hypervisor, virtualized block device, and metadata are alive

APP and os are dead

Hypervisor crash
Sego can’t trust OS journal

- Modern file systems use journals
- Journals have complex write ordering and recovery

**Challenges**
- Journal makes recovery easier for OS
- But more difficult for Sego!
- Hypervisor cannot trust OS
File length attack

Administrator

Configuration file

logical view of storage
File length attack

1. Append and close the file

Configuration file
Security setting

Correct file length

logical view of storage

Administrator

APP
OS
Sego hypervisor
Virtualized block device
File length attack

1. Append and close the file
2. Open the file

Correct file length

logical view of storage

Configuration file
Security setting

Administrator

APP
OS
Sego hypervisor
Virtualized block device
1. Append and close the file
2. Open the file

Correct file length

OS tells the length of previous state
File length attack

If the APP believes the OS length, OS can do the file length attack (undo the security setting)
File length attack

1. Append and close the file
2. Open the file

If the APP believes the OS length, OS can do the file length attack (undo the security setting)
Virtualized block device tracks file length with metadata

- Pervasive metadata model
- Metadata is shared

- Virtualized block device
- Tracks a maximum offset
- Shares the file length with hypervisor
Append crash scenario

APP
OS
Sego hypervisor
Virtualized block device

Secure file
Append crash scenario

APP
OS
Sego hypervisor
Virtual Offset block: 1000

Secure file
Append crash scenario

- Sego hypervisor
- Virtualized block device

<table>
<thead>
<tr>
<th>Secure file</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
</tr>
</tbody>
</table>
Append crash scenario

Write ordering by OS file system
Append crash scenario

Write ordering by OS file system

Offset 1000 (Data) → I-node (Journal) → I-node (Data)
Append crash scenario

Write ordering by OS file system

Offset 1000 (Data) -> I-node (Journal) -> I-node (Data)

Journaling filesystem discards the write during recovery
Append crash scenario

Write ordering by OS file system

Offset 1000 (Data) $\rightarrow$ I-node (Journal) $\rightarrow$ I-node (Data)

Journaling filesystem discards the write during recovery
Journal file system creates inconsistency problem
Journal file system creates inconsistency problem

APP
OS
Sego hypervisor
Virtualized block device

Secure file Offset 1000

Hypervisor’s length

OS’s length
Journal file system creates inconsistency problem

- APP
- OS
- Sego hypervisor
- Virtualized block device

Secure file
Offset 1000

Hypervisor’s length
OS’s length

Ask the file length
Journal file system creates inconsistency problem

- Secure file
- Offset 1000

Hypervisor's length

OS's length

Virtualized block device

Sego hypervisor

OS

APP

Ask the file length

Read offset 1000
Journal file system creates inconsistency problem

APP cannot progress
OS is not able to locate offset 1000
Read offset 1000

Hypervisor’s length

OS’s length

Secure file
Offset 1000

APP
OS
Sego hypervisor
Virtualized block device
Sego cannot trust journal file system

- This OS recovery is legal
  - Hypervisor cannot trust it
  - Legal or malicious?

- If APP believes OS’s length
  - OS can use this crash for the file length attack

- If APP believes hypervisor’s length
  - APP cannot progress in legal recovery case
Sego recovers secure file with metadata

Recovery procedure

- **APP**
  - Sego library
- **OS**
  - Sego hypervisor
  - Virtualized block device
- **Metadata**
  - Offset
  - Sector number
- **Secure file**
  - Offset 1000

Hypervisor’s length
OS’s length
Recovery procedure

**APP**
- Opens the file
- Get OS length

**LIB**

**Metadata**
- Offset
- Sector number

**Offset 1000**

**Secure file**

**Hypervisor’s length**

**OS’s length**

**Sego hypervisor**

**Virtualized block device**

**OS**

**Sego library**

**APP**
Sego recovers secure file with metadata

Recovery procedure

**APP**
- Opens the file

**LIB**
- Get OS length
- Give OS length to Sego hypervisor

**Sego library**
- Offset
- Sector number

**OS**
- Hypervisor’s length
- OS’s length

**Sego hypervisor**

**Virtualized block device**

**Secure file**
- Offset 1000

**Metadata**
- ...

**Hypervisor’s length**

**OS’s length**
Sego recovers secure file with metadata

**Recovery procedure**

**APP**
- Opens the file
- Get OS length

**LIB**
- Give OS length to Sego hypervisor

**Length mismatch**

**Metadata**
- Offset
- Sector number

**Offset**
- Sector number

**Sego hypervisor**

**Virtualized block device**

**Sego library**

**OS**

**Secure file**

**Hypervisor’s length**

**OS’s length**

**APP**

**LIB**
Sego recovers secure file with metadata

Recovery procedure

**APP**
- Opens the file
- Get OS length

**LIB**
- Give OS length to Sego hypervisor

**LIB**
- Length mismatch

**HYP**
- Requests recovery of offset 1000

**APP**
- Sego library

**OS**
- Sego hypervisor

**Virtualized block device**

**Secure file**
- Offset 1000

**Hypervisor’s length**

**OS’s length**

Metadata

- Offset
- Sector number

Length mismatch
Recovery procedure

**APP**
- Opens the file
- Get OS length

**LIB**
- Get OS length to Sego hypervisor
- Length mismatch

**HYP**
- Requests recovery of offset 1000

**Virtualized block device**
- Read the data by looking up metadata

**Sego library**

**OS**

**Sego hypervisor**

**Virtualized block device**

**Metadata**
- Offset
- Sector number
- ...

**Secure file**
- Offset 1000

**Hypervisor’s length**

**OS’s length**
Recovery procedure

**APP**
- Opens the file

**LIB**
- Get OS length
- Give OS length to Sego hypervisor

**HYP**
- Requests recovery of offset 1000

**Virtualized block device**
- Read the data by looking up metadata
- Copies the offset 1000 to the library

**Metadata**
- Offset
- Sector number

**Offset 1000**

**Secure file**

**Hypervisor’s length**

**OS’s length**
Sego recovers secure file with metadata

Recovery procedure

**APP**
- Opens the file
- Get OS length

**LIB**
- Give OS length to Sego hypervisor

**LIB**
- Length mismatch

**HYP**
- Requests recovery of offset 1000

**Virtualized block device**
- Read the data by looking up metadata
- Copies the offset 1000 to the library

**Metadata**
- Offset
- Sector number

**OS**
- Sego hypervisor

**Secure file**
- Offset 1000

**Hypervisor’s length**

**OS’s length**

*Copies the offset 1000 to the library*
Sego recovers secure file with metadata

Recovery procedure

APP
- Opens the file
- Get OS length

LIB
- Get OS length
to Sego hypervisor
- Length mismatch

LIB
- Give OS length
to Sego hypervisor

HYP
- Requests recovery of offset 1000
- Virtualized block device
  - Read the data by looking up metadata
  - Copies the offset 1000 to the library

Sego library

OS

Sego hypervisor

Virtualized block device

Metadata
- Offset
- Sector number
- ...

Secure file
- Offset 1000

Hypervisor’s length
- OS’s length
Recovery procedure

**APP**
- Opens the file
- Get OS length

**LIB**
- Get OS length to Sego hypervisor
- Length mismatch

**HYP**
- Requests recovery of offset 1000

**Virtualized block device**
- Read the data by looking up metadata
- Copies the offset 1000 to the library

**Sego library**
- Sego hypervisor
- Virtualized block device

**Metadata**
- Offset
- Sector number

**Secure file**
- Offset 1000

**Hypervisor’s length**

**OS’s length**
## Other crash cases

<table>
<thead>
<tr>
<th>Recovery target</th>
<th>Inconsistency</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>File creation</td>
<td>File is created in hypervisor but not in OS</td>
<td>When the APP opens the file</td>
</tr>
<tr>
<td>File length</td>
<td>File length of hypervisor and OS is different</td>
<td>When OS reboots from crash</td>
</tr>
<tr>
<td>Data recovery</td>
<td>Hypervisor loses blocks because OS discards them</td>
<td>When the APP opens the file</td>
</tr>
<tr>
<td>Block commit (hypervisor crash)</td>
<td>Block write might not be committed in virtual block</td>
<td>Hypervisor runs FSCK</td>
</tr>
<tr>
<td>Crash while recovery</td>
<td>One of the above</td>
<td>Hypervisor runs FSCK</td>
</tr>
</tbody>
</table>
Fault injection

• Fault injector
  • Modify previous framework for modern OS
    • Nooks (Swift et al., SOSP 2003)
    • Rio file cache (Chen et al., ASPLOS 1996)
  • Fault distribution is based on real-world fault study
    • An empirical study of operating system error (SOSP 2001)
    • Faults in linux: Ten years later (ASPLOS 2011)
    • A study of linux file system evolution (FAST 2013)
## Crash recovery experiment

<table>
<thead>
<tr>
<th>Recovery</th>
<th>4 writing processes</th>
<th>Git</th>
</tr>
</thead>
<tbody>
<tr>
<td>No crash</td>
<td>51 (51%)</td>
<td>114 (76%)</td>
</tr>
<tr>
<td>File creation</td>
<td>40 (40%)</td>
<td>29 (19%)</td>
</tr>
<tr>
<td>File length</td>
<td>2 (2%)</td>
<td>7 (5%)</td>
</tr>
<tr>
<td>Data Recovery</td>
<td>1 (1%)</td>
<td>0</td>
</tr>
</tbody>
</table>

- Experiment
  - 4 processes write each secure file and verify them
  - Git: add files (20MB), sync, and add files (30MB).
  - 20 randomly selected faults are injected

Without Sego’s recovery Application keeps crashing

Sego correctly recovers every case
# Sego overhead

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Slowdown to Linux-VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenLDAP</td>
<td>Insert (15.9%), Query (3.6%), Delete (15.0%)</td>
</tr>
<tr>
<td>Apache</td>
<td>Throughput (7.5%), Latency (8.2%)</td>
</tr>
<tr>
<td>Grep</td>
<td>Small file (10.1%), Large file (8.3%)</td>
</tr>
<tr>
<td>DokuWiki</td>
<td>90/10 read/write web pages (49%)</td>
</tr>
</tbody>
</table>
Conclusions

• Sego proposes the pervasive metadata model for
  • eliminating encryption and hashing for performance without losing security guarantees
  • detecting file system inconsistencies and recovery from crashes
• We hope the trusted metadata model will be adapted to device virtualization
Questions?