

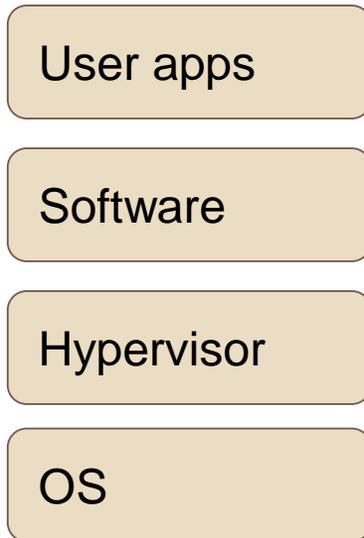
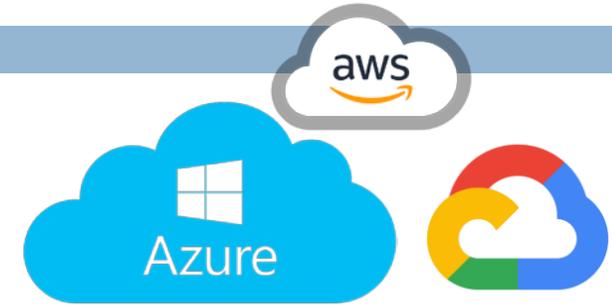
# MODERN SYSTEMS: SECURITY

CS6410

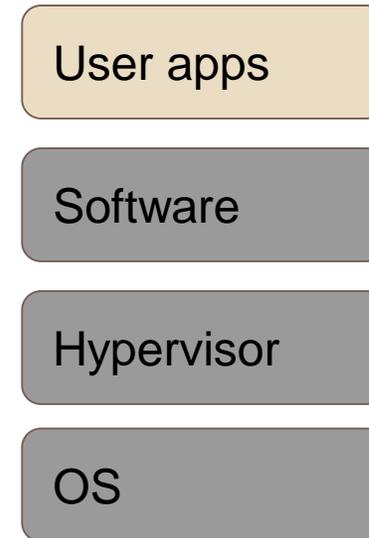
Hakim Weatherspoon

Slides borrowed liberally from past presentations from Sai Krishna Deepak Maram

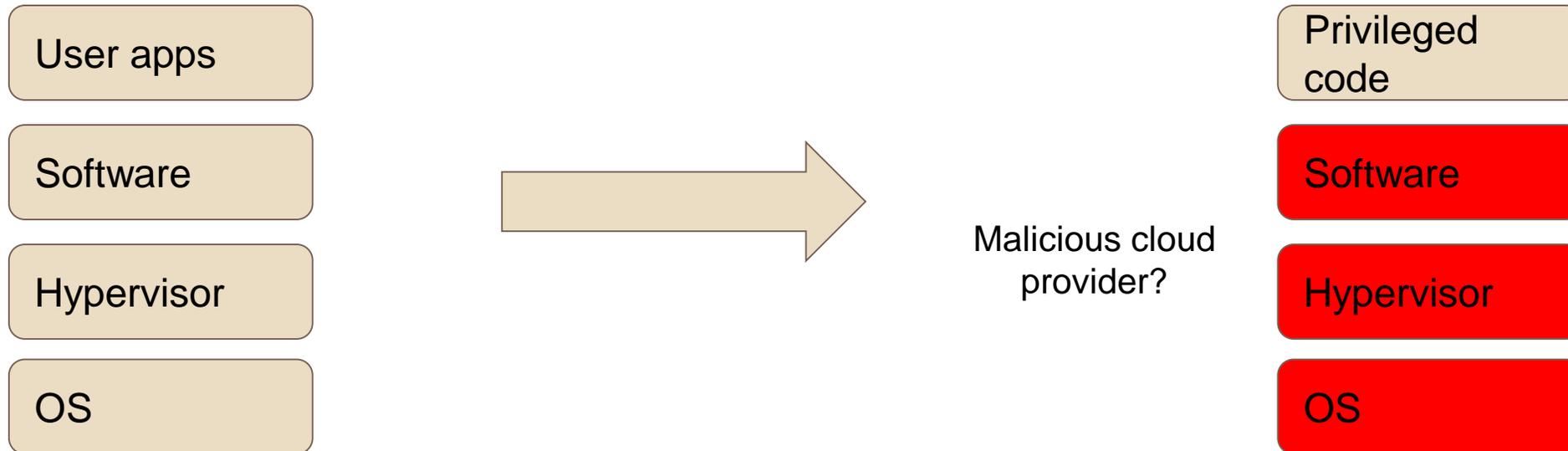
# Move to a Cloud-based model



**PaaS**  
Cloud Provider  
manages the  
stack



# Move to a Cloud-based model



# Can you trust the cloud?

---

- ❑ Huge Trusted Computing Base (TCB)
- ❑ Cloud Provider's software
- ❑ Management stack
- ❑ Sysadmins



What do we want?

—

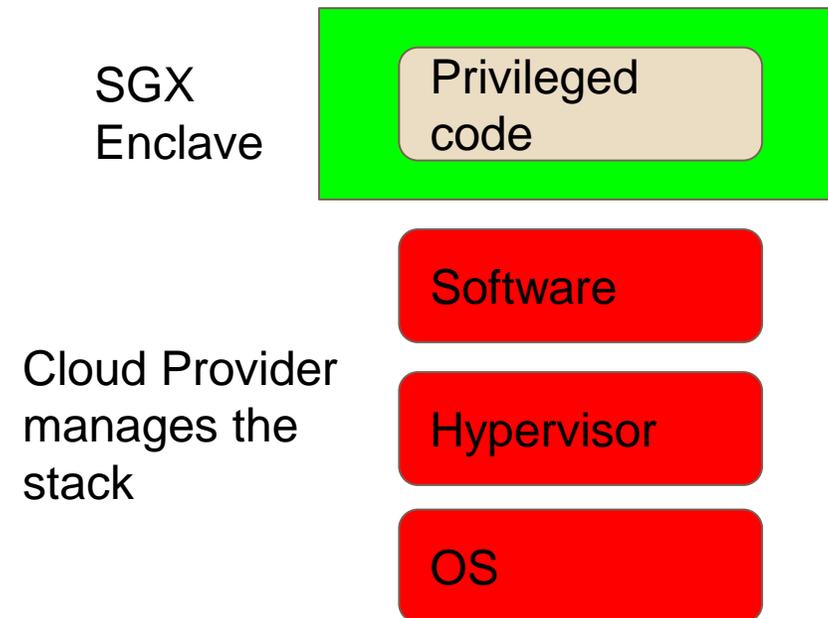
# Shielded Execution using SGX

## □ Confidentiality

- ▣ The execution state is unobservable to the rest of the system.

## □ Integrity

- ▣ If the program completes, its output is the same as a correct execution on a reference platform



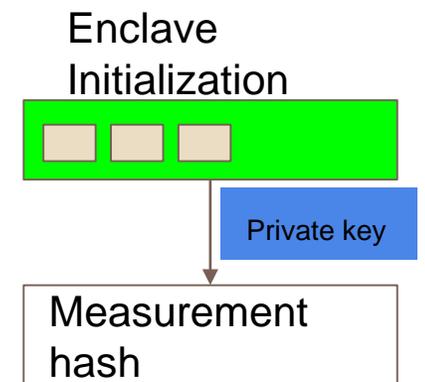
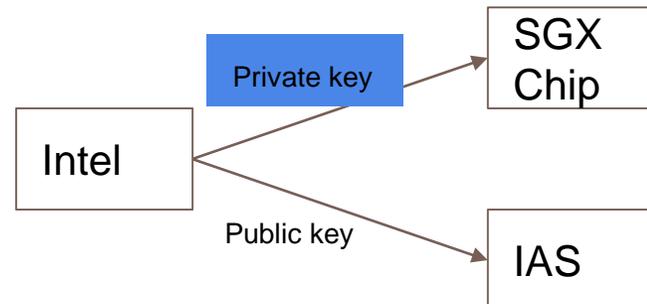


Is shielded execution sufficient?

—

# Remote Attestation

- **Goal:** Allow cryptographic verification that specific software has been loaded within an enclave
  - ▣ While an enclave is initialized, its contents is cryptographically hashed by the CPU forming the enclave's *measurement*
- Generated using a key burnt on the SGX chip
  - ▣ Root of trust: Intel
  - ▣ Intel attestation service (IAS) for verification





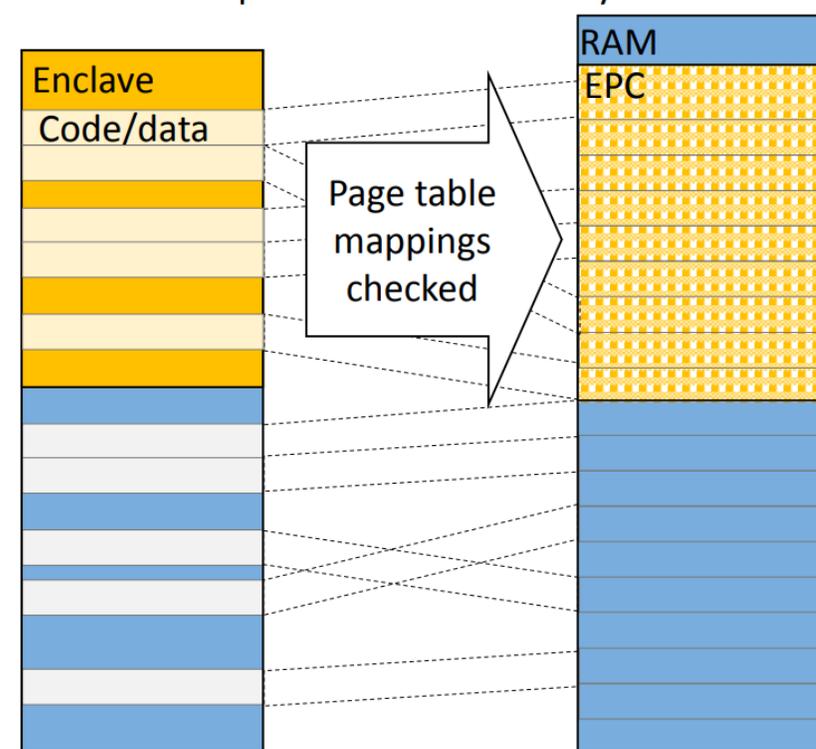
How does SGX achieve this?

—

# Memory protection

- EPC (Enclave Page Cache)
  - A separate region in physical memory
  - All enclave pages reside here
  - Hardware tracks meta info corresponding to each page
  - Virtualized

Virtual address space



# Memory protection

- EPC (Enclave Page Cache): A separate region in physical memory
  - ▣ Encrypted and integrity-protected before writing to the main memory
- Same page table as the underlying OS
  - ▣ **Access checks** are performed to ensure any other application (not even other enclaves) can access an enclave's data

# Execution lifecycle (high-level)

- Loading stage: Performed by untrusted code
  - ▣ Enclave is initialized by copying code/data into **EPC Pages**
  - ▣ At the end of which, contents are hashed to compute enclave's **measurement hash**
- Enclave mode:
  - ▣ Special instructions to create an enclave, add pages to enclave and exit an enclave
  - ▣ Similar to switching from user to kernel mode
  - ▣ Secure mechanisms to handle interrupts (or) page faults to protect from OS exception handlers



Before SGX?

—

# Trusted Platform Module (TPM)

- Attestation-based
- Can be used with commodity systems
- **Weak security**
  - ▣ Much bigger TCB than SGX: Measurement hash covers all the OS modules and device drivers
  - ▣ Very hard to keep an up-to-date list of the hashes
  - ▣ Many more attacks....



# How to port legacy applications into SGX?

—

# Developing applications in SGX

- Untrusting OS: Makes it harder
- Any function call (or) syscall made outside the enclave are not guaranteed to return
- Even if data returns, enclave cannot trust the data returned

# Haven

- Haven design goals:
  - ▣ Mutual distrust b/w guest and host
  - ▣ Run legacy apps inside SGX without any modifications
- Application interacts only with LibOS
  - ▣ Assumes libOS is carefully implemented
- Shield module interacts with the untrusted host OS

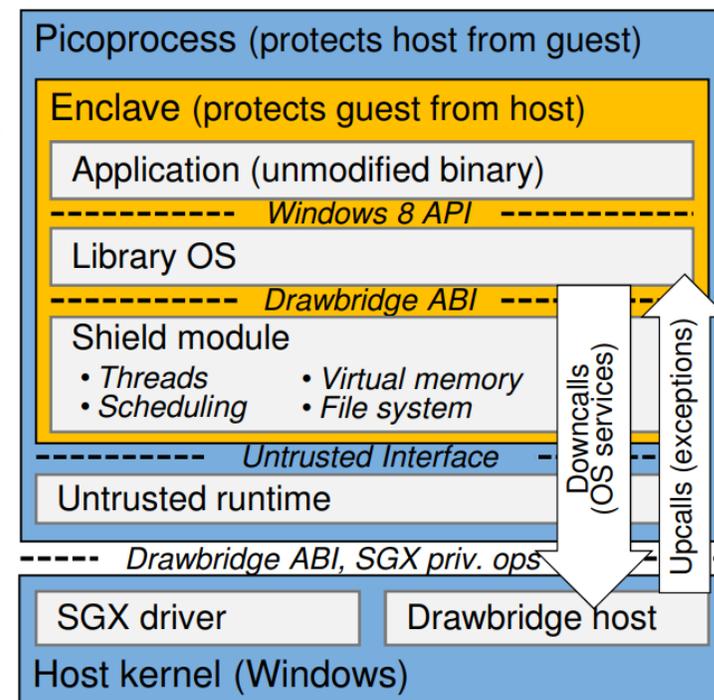


Figure 2: Haven components and interfaces

# How Haven handles ligo attacks

- ligo attacks: “Malicious kernel attempts to subvert an isolated application by exploiting its assumption of correct OS behaviour, for example when using the results of system calls”
- **LibOS:** Implement entire OS as part of the Trusted Computing Base. Limits the interaction of enclave app with the actual OS, thus reducing the attack surface.

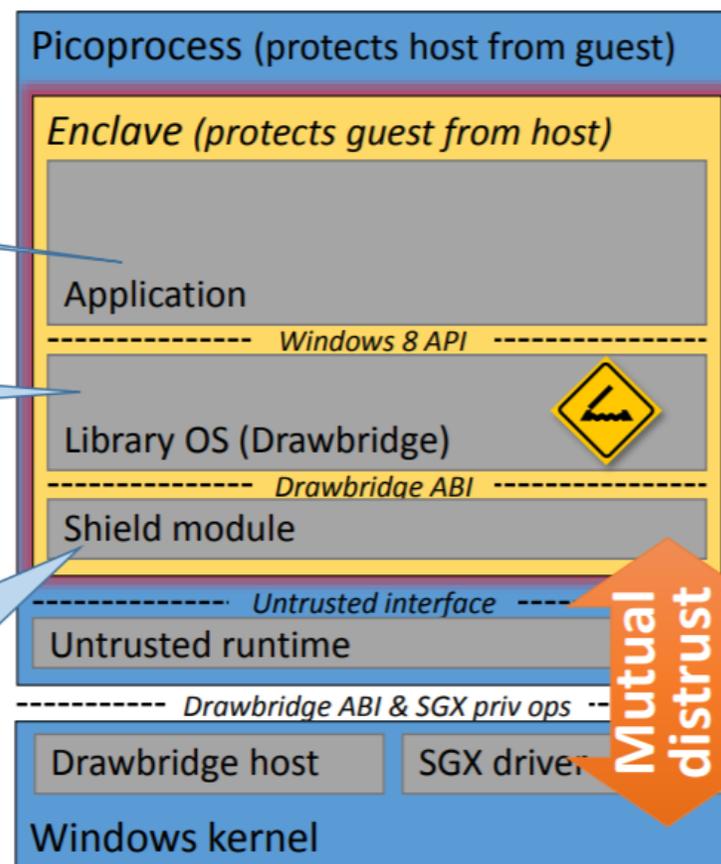
# Haven

## Haven

- Unmodified binaries

- Subset of Windows, enlightened to run in-process

- Shields LibOS from Iago attacks
- Includes typical kernel functionality
  - Scheduling, VM, file system
- Untrusted interface with host



# LibOS and Exokernels

Both bring OS level functionalities to the user space, but for what reasons?

- Efficiency in Exokernel
  - ▣ “Move OS functionality to the user space to grant more flexibility”
- Security in Haven’s LibOS
  - ▣ “Move OS functionality into the enclave to reduce attack space”

# Haven Performance

---

- 35% - 65% slowdown
- Depends on the exact use case



# Haven influencing SGX

—

# Haven influencing SGX design

- Dynamic memory allocation
  - ▣ SGX does not allow addition of enclave pages after the creation of enclave
- Exception Handling
  - ▣ SGX does not allow handling of all exceptions
- Some other limitations
  
- **Fixed in v2.0**

# SGX: What's New?

---

- Latest v2.3
  - ▣ Trusted randomness, other crypto operations
  - ▣ File abstractions inside an enclave
- Baidu's Rust SGX SDK
  - ▣ Dockerized
  - ▣ Runs a simulated version on machines without SGX chip as well

# Is SGX Secure?

---

- Sophisticated side channel attacks
  - ▣ Foreshadow - Usenix'18
  - ▣ Speculative execution

Trusted hardware  
makes the attacker's  
job costly



## FORESHADOW

Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution

# Discussion

---

## □ Haven

- ▣ Exokernel connection to Haven
- ▣ Impact of Haven and why it's not more widely used?

## □ SGX

- ▣ Does trusted hardware solve the problem of security in the cloud?
- ▣ Can SGX still be useful in face of side channel attacks?

Thank you!

References:

1. [Haven, Slides](#)
2. [Intel SGX explained](#)





**Thank You!**

# Perspective

- Virtualization: creating a illusion of something
- Virtualization is a principle approach in system design
  - ▣ OS is virtualizing CPU, memory, I/O ...
  - ▣ VMM is virtualizing the whole architecture
  - ▣ What else? What next?

# Next Time

- Project: next step is the Survey Paper due next Friday
- Read and write a review:
  - ▣ ***The Google file system***, Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung. *19th ACM symposium on Operating systems principles (SOSP)*, October 2003, 29--43.
  - ▣ ***Spanner: Google's Globally Distributed Database***, James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. *In Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI'12)*, October 2012, 251--264.