SoK: Introspections on Trust and the Semantic Gap

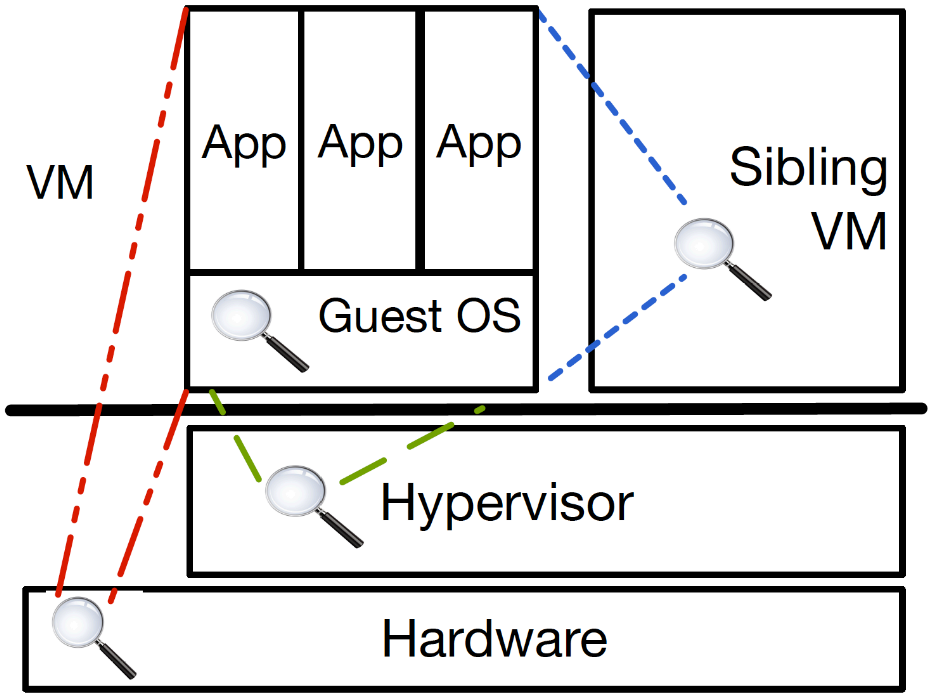
Presented by Zhenyu Ning

Contents

Contents

VMI

- Virtual Machine Introspection
  - Memory, disk, network traffic
  - Smaller TCB and less CVEs
- A monitor tracks the behavior of guest OS.
  - Hypervisor, sibling VM, guest OS, hardware



Semantic Gap

- The gap between high-level expressions and hardware-level abstractions.

```
struct task_struct {
    volatile long state;    /* -1 unrunr
    void *stack;
    atomic_t usage;
    unsigned int flags; /* per process f
    unsigned int ptrace;

#ifdef CONFIG_SMP
    struct llist_node wake_entry;
    int on_cpu;
    struct task_struct *last_wakee;
    unsigned long wakee_flips;
    unsigned long wakee_flip_decay_ts;
```

=?

```
0x00000001  0x00000000  0x77CD8000  0xFFFFFFC9  0x00000002  0x00400100  0x00000000  0x00000000
0x00000000  0x00000000  0x00000000  0x00000000  0x774B1180  0xFFFFFFC9  0x00000005  0x00000000
0xFFFF9F48  0x00000000  0x00000000  0x00000000  0x00000078  0x00000078  0x00000078  0x00000000
0x00895388  0xFFFFFFC0  0x00000400  0x00000000  0x00400000  0x00000000  0x74FDB239  0xFFFFFFC9
0x5D11C838  0xFFFFFFC9  0x00000000  0x00000000  0x77CD6C90  0xFFFFFFC9  0x77CD6C90  0xFFFFFFC9
0x00000000  0x00000000  0xB2F71A20  0x0000000C  0x9ED87DEC  0x00000000  0x17CAEB3B  0x00000004
0x9ED64860  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x098DDCF0  0x00000000
0x000003D9  0x00000000  0x3A21FF04  0x00000000  0x00000019  0x00000000  0x291F3BB8  0x00000000
0xB2F71A20  0x0000000C  0x3C752F94  0x00000003  0xD809E870  0x0000000B  0x00000000  0x00000000
0x25B2AF14  0x00000000  0x049EAE90  0x00000000  0x15912610  0x00000000  0x00000000  0x00000000
0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000
0x000000A5  0x00000000  0x00000026  0x00000000  0x00000000  0x00000000  0x000000A5  0x00000000
```

Contents

Bridging the gap

- Learning and reconstruction
- Code implanting
- Process Outgrafting

Learning and reconstruction

- Learning phase
  - Generate data structure signature
- Search phase
  - Identify the instance of data structure in memory

Hand-craft data structure signature

- Based on expert knowledge of the internal workings of an OS.
- Example: find “init_task”, then go through the linked list.
- Disadvantage: Inflexible

Source code analysis

- Based on analysis of source code.
- Leverage static analysis to generate a graph of kernel data structures.
- Challenge: Invalid pointer, object pools.

Dynamic learning

- Based on dynamic analysis of an OS instance.

- Training on a trusted OS instance by manipulate a data structure of interest.

- Robust signature.

Search phase

- **Linearly Scanning**
  - Access more memory
  - Immune to broken pointers
- **Pointer traversing**
  - Traverse less total memory
  - Suffer from cyclic and invalid pointers
- **Large overhead leads to low frequency.**

Code implanting

- Implanting the monitor code into guest OS.
  - Implant process
  - Implant function

- Challenge: Integrity of implanted code and guest kernel.

Process outgrafting

- Monitor a untrusted VM from another sibling trusted VM.
- The trusted VM has some visibility into the kernel memory of untrusted VM.
- Using existing code and read-only heap

Kernel executable integrity

- W XOR X mechanism
- Whitelist
- Protect object hooks

Control Flow Integrity(CFI)

Contents

1. Background
2. Bridge semantic gap
3. Design choices
4. Attacks and defense
5. Bridge semantic gap, again
6. Future work & Conclusion

Prevention & detection

- Detection
  - Identify violation of security policy
  - Issue: recovery
- Prevention
  - Detection and interposition
  - Issue: performance overhead

Asynchronous & synchronous

- Synchronous
  - Prevention system, high overhead
- Asynchronous
  - Introspect into a snapshot of memory
- Trade-offs across performance & risk
- Assumption: Knowing all hook location, object slab

Snapshotting & Snooping

- Snapshotting
  - Use PCI device to take RAM snapshots
  - Together with value of CPU register
  - SMM-based solution
  - Suffer from DOS attack

Snapshotting & Snooping

- Snooping
  - Lightweight hardware
  - Monitor writes to sensitive code region and detect updates to memory from malicious device or driver by DMA
  - Use snapshotting device to check data structure invariants or code integrity
  - Do not use commodity hardware and only focused on detection

Contents

1. Background
2. Bridge semantic gap
3. Design choices
4. Attacks and defense
5. Bridge semantic gap, again
6. Future work & Conclusion

KOH

- Kernel Object Hooking(KOH)
  - Modify function pointers in kernel text or data section
  - Example: override readdir()
- Text section hook
  - W XOR X mechanism
- Data section hook
  - Move hooks or whitelist
- Assumption: benign kernel, ability of administrator

DKOM

- Dynamic Kernel Object Manipulation(DKOM)
  - Modify kernel heap
  - Example: remove process from double linked list
- Detect data structure invariant violation asynchronously
- Assumption
  - Have found all security-relevant data structures
  - These structures all have invariants
  - Detector will win the race

DKSM

- Direct Kernel Structure Manipulation(DKSM)
  - Change interpretation of data structure
  - Different interpretation between training and classification

- Precluded by a generous threat model

Contents

1. Background
2. Bridge semantic gap
3. Design choices
4. Attacks and defense
5. Bridge semantic gap, again
6. Future work & Conclusion

Semantic gap

- **Weak semantic gap**
  - An solved engineering challenge
  - Assume guest OS is benign during training and won't have different behavior under monitoring
- **Strong semantic gap**
  - An open security problem
  - Do not make any assumption about the guest OS

Semantic gap

- Paraverification
  - Light modification to guest OS
  - guest OS provide evidence of its action is correct
- Hardware support
  - Hardware-assisted memory isolation, like SGX
- Reconstruction from untrusted sources
  - Incrementally training
  - Inconsistency detection

Contents

1. Background
2. Bridge semantic gap
3. Design choices
4. Attacks and defense
5. Bridge semantic gap, again
6. Future work & Conclusion

Future work

- Scalability
  - Overhead not acceptable in multi-VM system
  - Balance of overhead and risk
- Privacy
  - evaluate risks of new side channels

Conclusion

- Researches should be refocused on removing the assumptions of a guest OS to reduce the TCB

- Future solutions should pay more attention to scalability and privacy concerns

Reference

- Jain B, Baig M B, Zhang D, et al. Sok: Introspections on trust and the semantic gap[C]//Security and Privacy (SP), 2014 IEEE Symposium on. IEEE, 2014: 605-620.

Thank you!