Paper Review

Tony Espinoza

am.espinoza@utexas.edu

ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds

- Summarize

Multi-tenancy cloud computing

- What is it?
- What is the potential threat?
- What dose this look like?
- Why is it used?

Virtual Machines

- The older method was to use VMs.
- Were there still threats here?
  - Hey, You, Get Off of My Cloud (ref 35).
- Are the threat models the same?

Multi-tenancy

- The issue with multi-tenancy?
- Not all subsystems in Linux can tell the difference between the container and host.
  - This could possibly expose system-wide info to containerized apps.
  - Why is this bad?

Side channel

- What is a side channel
  - Any channel you can use to infer/transfer data.
  - Shared, limited resource.
  - Examples:
    - SYN cache (Network).
    - Drive RW speed.
    - Power consumption.

Possible channels

- Two groups of information channels.
  - Host system.
  - Individual process execution.

Possible channels?

- Host system information
  - Performance data.
  - Global kernel data.
  - Asynchronous kernel events.
  - Power consumption.

Possible channels?

- Individual process execution information.
  - Process scheduling.
  - cgroups.
  - Process running status.

Testing

- Docker
- LinuX Contaier (LXC)
  - First complete Linux container manager (2008).

Background

- Namespaces:
  - Isolate view of what is in the namespace.
  - MNT, UTS, PID, NET, IPC, USER, CGROUP.
- Cgroups:
  - Resource limit.

Why is a power attack possible

- Data centers host more machines than they can handle at peak power.
  - Peak power in never really achieved.
  - Same reason airlines overbook flights.
    - Statisticly not everyone will show.
    - Statisticly not all machines will require peak power simultaneously.

Anatomy of a power attack

- Attacker needs:
  - Access to servers in the target data center.
  - Steadily running moderate workloads to increase the power consumption of servers.
  - To abruptly switch to power-intensive workloads to trigger power spikes.
- This can cause a power spike and a circuit to be tripped.
- Servers should run on same rack to maximize the attack.

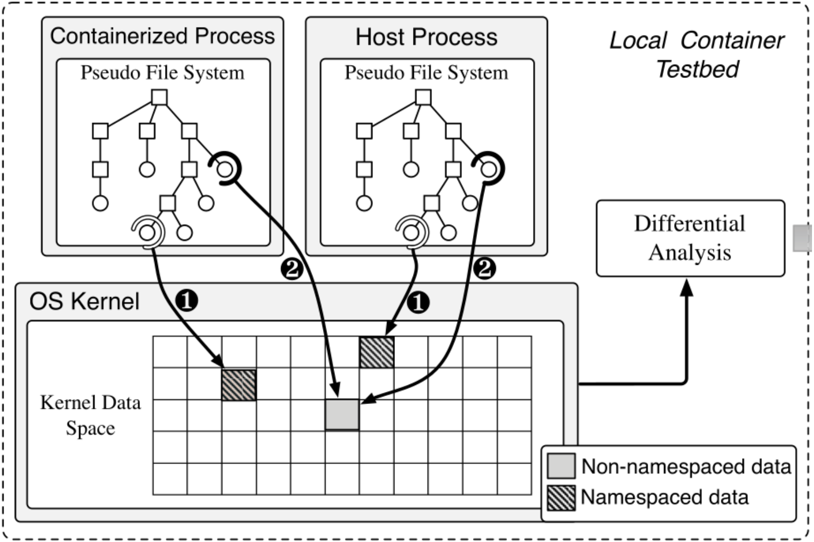
Container information leakages

- Two interfaces for leaks.
  - System calls.
  - Pseudo file systems.
    - /proc
    - /sys
- Which does the paper use?

Pseudo file systems

- How do we leverage them?
- Compare pseudo file system of:
  - Containerized.
  - Host process.

Comparing pseudo file systems



Inference of co-resident container

- Why is co-residence bad?
  - Can hijack user accounts.
  - Extract private keys.
- How to tell if you are co-resident

Co-location checker

- This paper uses what attributes to test for co-location?
  - Uniqueness $\mathbb{U}$.
    - Can the channel uniquely id a host?
  - Variation $\mathbb{V}$.
    - Test the variation of a file over time and compare.
  - Manipulation $\mathbb{M}$.
    - Manipulate data.

Comparing pseudo file systems

TABLE II: LEAKAGE CHANNELS FOR CO-RESIDENCE VERIFICATION.

Leakage Channels	𝕌	𝕍	𝕄	Rank
/proc/sys/kernel/random/boot_id	●	○	○	█████
/sys/fs/cgroup/net_prio/net_prio.ifpriomap	●	○	○	█████
/proc/sched_debug	●	●	●	█████
/proc/timer_list	●	●	●	█████
/proc/locks	●	●	●	█████
/proc/uptime	●	●	◐	████
/proc/stat	●	●	◐	████
/proc/schedstat	●	●	◐	████
/proc/softirqs	●	●	◐	████
/proc/interrupts	●	●	◐	████
/sys/devices/system/node/node#/numastat	●	●	◐	████
/sys/class/powercap/.../energy_uj[2]	●	●	◐	████
/sys/devices/system/.../usage[3]	●	●	◐	████
/sys/devices/system/.../time[4]	●	●	◐	████
/proc/sys/fs/dentry-state	●	●	◐	███
/proc/sys/fs/inode-nr	●	●	◐	███
/proc/sys/fs/file-nr	●	●	◐	███
/proc/zoneinfo	○	●	◐	██
/proc/meminfo	○	●	◐	██
/proc/fs/ext4/sda#/mb_groups	○	●	◐	██
/sys/devices/system/node/node#/vmstat	○	●	◐	██
/sys/devices/system/node/node#/meminfo	○	●	◐	██
/sys/devices/platform/.../temp#_input[5]	○	●	◐	█
/proc/loadavg	○	●	◐	█
/proc/sys/kernel/random/entropy_avail	○	●	◐	█
/proc/sys/kernel/.../max_newidle_lb_cost[6]	○	●	○	█
/proc/modules	○	○	○	▯
/proc/cpuinfo	○	○	○	▯
/proc/version	○	○	○	▯

Monitor power consumption

- Use Running Average Power Limit (RAPL).
- /sys/class/powercap/intel-rapl.
- Accessible to containers.
- System wide power info of host:
  - core
  - DRAM
  - package

Goal of information leak

- What is the goal of finding these information leaks?
- What do we want to do with the information?
  - Infer co-location.
  - Monitor power consumption.

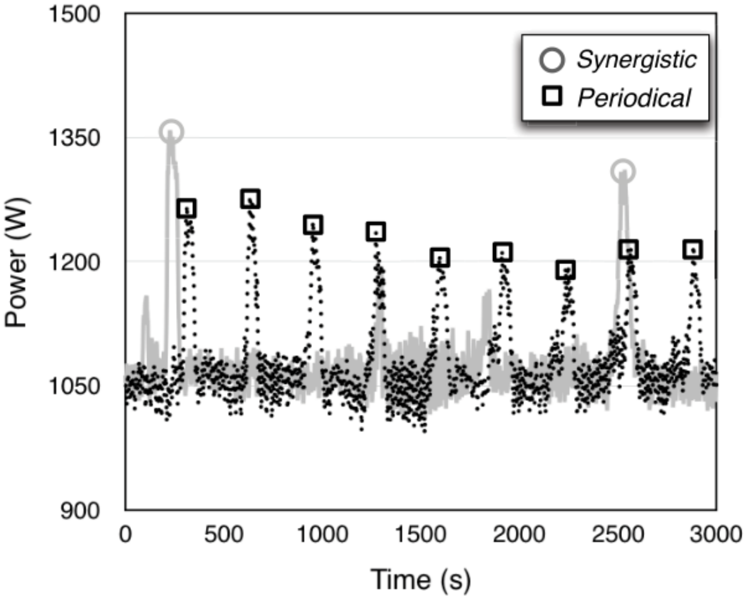
Put it all together

- What can we do with:
  - Co-located containers.
  - Power spike attacks.
  - Knowledge of power consumption.
- Synergistic power attacks.

Amplify attack

- Monitor the power.
- Learn when peak power consumption is.
- Attack at peak power consumption time.

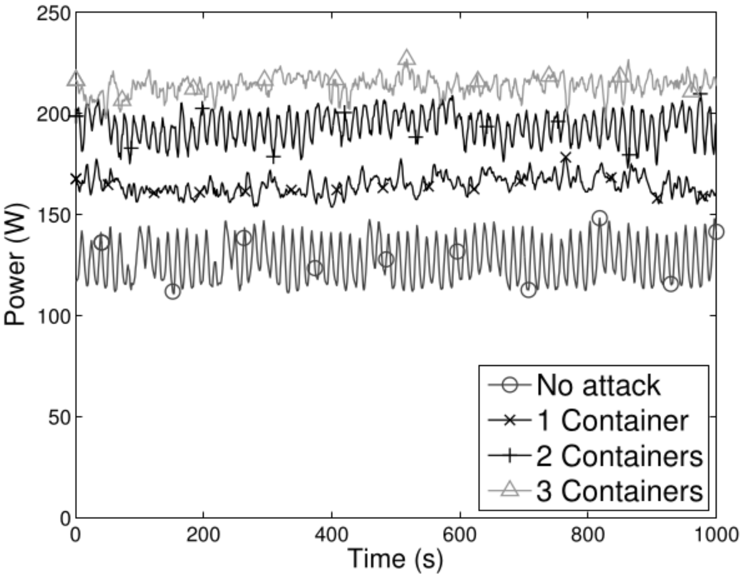
Amplify attack



Attack Orchestration

- If the attack is launched from the same machine we can make a bigger power spike.
- Create containers
  - Check for co-location
  - Repeat.
- Run prime benchmark.

Attack Orchestration



Defences

- Two stage defence:
  - Masking the side channels.
  - Enhancing the container's resource isolation.

Masking side channels.

- Make pseudo file systems unreadable.
  - What could you use to do this easily?
    - SELinux.
    - AppArmor $<$- They chose this one.

Power-based Namespace

- The authors add a power-based namespace.
- Use the RAPL interface for each container.
  - Accurate
    - Need a software-based modeling.
  - Efficient
    - Want minimal overhead.

Power consumption

- Accumulated energy usage for:
  - Package.
    - $M_{Core} + M_{dram} + \lambda$
  - Core.
    - $F(CM/C, BM/C) * I + \alpha$
  - DRAM
    - $\beta * CM + \gamma$

CM = cache misses, BM = branch misses, C = CPU cycles. I = # retired instructions $\alpha\beta\gamma$ are derived constraints.

Defence performance

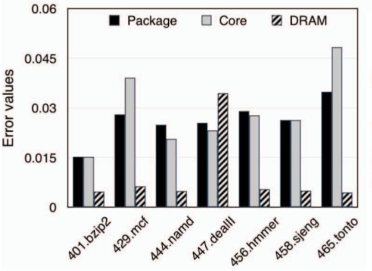


Fig. 8: The accuracy of our energy modeling approach to estimate the active power for the container from aggregate event usage and RAPL.

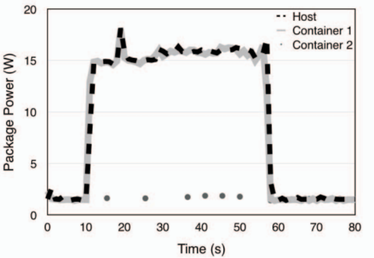


Fig. 9: Transparency: a malicious container (Container 2) is unaware of the power condition for the host.

Defence performance

TABLE III: PERFORMANCE RESULTS OF UNIX BENCHMARKS.

Benchmarks	1 Parallel Copy			8 Parallel Copies		
	Original	**Modified**	**Overhead**	**Original**	**Modified**	**Overhead**
Dhrystone 2 using register variables	3,788.9	3,759.2	0.78%	19,132.9	19,149.2	0.08%
Double-Precision Whetstone	926.8	918.0	0.94%	6,630.7	6,620.6	0.15%
Execl Throughput	290.9	271.9	6.53%	7,975.2	7,298.1	8.49%
File Copy 1024 bufsize 2000 maxblocks	3,495.1	3,469.3	0.73%	3,104.9	2,659.7	14.33%
File Copy 256 bufsize 500 maxblocks	2,208.5	2,175.1	0.04%	1,982.9	1,622.2	18.19%
File Copy 4096 bufsize 8000 maxblocks	5,695.1	5,829.9	-2.34%	6,641.3	5,822.7	12.32%
Pipe Throughput	1,899.4	1,878.4	1.1%	9,507.2	9,491.1	0.16%
Pipe-based Context Switching	653.0	251.2	61.53%	5,266.7	5,180.7	1.63%
Process Creation	1416.5	1289.7	8.95%	6618.5	6063.8	8.38%
Shell Scripts (1 concurrent)	3,660.4	3,548.0	3.07%	16,909.7	16,404.2	2.98%
Shell Scripts (8 concurrent)	11,621.0	11,249.1	3.2%	15,721.1	15,589.2	0.83%
System Call Overhead	1,226.6	1,212.2	1.17%	5,689.4	5,648.1	0.72%
System Benchmarks Index Score	2,000.8	1,807.4	9.66%	7,239.8	6,813.5	7.03%

Thoughts

- Fundamental or artifactual?
  - What is the main problem?
  - What was the root cause of the issue?
- Evaluation?