Hprobes: Dynamic VM Monitoring

Z. J. Estrada, C. Pham, Z. Kalbarczyk, R. K. Iyer

ACC Meeting 2014-09-24
Monitoring Techniques

• Passive
Monitoring Techniques

• Passive
  ○ Polling
Monitoring Techniques

- Passive
  - Polling
  - “Traditional”
Monitoring Techniques

• Passive
  ○ Polling
  ○ “Traditional”

• Active
Monitoring Techniques

- Passive
  - Polling
  - “Traditional”
- Active
  - Event-driven
Monitoring Techniques

• Passive
  ◦ Polling
  ◦ “Traditional”

• Active
  ◦ Event-driven
  ◦ HyperTap (and others...)
Active Monitoring: Hooks

Guest VM

Untrusted VM

Trusted VM

Hypervisor

Hypervisor

Zak Estrada

Hprobes
Figure 1: Formal model of secure active monitoring shown with potential attacks.

Properties of Lares, SIM, ...

+ Active monitoring
Properties of Lares, SIM, ...

+ Active monitoring
+ Protected hooks
Properties of Lares, SIM, ...

+ Active monitoring
+ Protected hooks
- Not transparent - OS driver, etc...
Properties of Lares, SIM, ...

+ Active monitoring
+ Protected hooks
- Not transparent - OS driver, etc...
- Not dynamic - boot time config
Hprobes

- Active monitoring
Hprobes

- Active monitoring
- Transparent
Hprobes

• Active monitoring
• Transparent - OS agnostic
Hprobes

- Active monitoring
- Transparent - OS agnostic
- Dynamic
Hprobes

- Active monitoring
- Transparent - OS agnostic
- Dynamic
- Protected
Hprobes

- Active monitoring
- Transparent - OS agnostic
- Dynamic
- Protected
- Simple and Fast
Architecture

User processes

- Security Applications
- Hprobe user agent

Set/remove probes

Hprobe
Kernel agent

ioctl(…)

VM

- Probe
- Probe
- Probe

Event Forwarder

KVM Hypervisor

Linux kernel

- Set/Remove probes
- Set single step

Helper APIs

Zak Estrada
Hitting a probe

VM

probe hit
(int 3)
execute instr
trap
...

Hypervisor

Reset orig. inst
set single stepping
rewrite int3
resume

Zak Estrada
Hprobes

9/22
Where we are now

☑ Kernel-space probes
Where we are now

✓ Kernel-space probes
✓ Kernel patch
Where we are now

✓ Kernel-space probes
✓ Kernel patch
✓ CLI Useragent
Where we are now

✔ Kernel-space probes
✔ Kernel patch
✔ CLI Useragent
✔ integration with libvmi
Where we are now

- ✔ Kernel-space probes
- ✔ Kernel patch
- ✔ CLI Useragent
  - ✔ integration with libvmi
- ➡ User-space probes
Where we are now

- Kernel-space probes
- Kernel patch
- CLI Useragent
  - integration with libvmi
- User-space probes
  - Then actual applications
Potential Uses

What can we do with this?
Potential Uses

What can we do with this?

• Arbitrarily specify events
• More app-level monitoring
Potential Uses

- Heartbeat/watchdog
  - App-level
Potential Uses

Heartbeat/watchdog
- App-level
- Critical code sections
Potential Uses

Heartbeat/watchdog
- App-level
- Critical code sections
- Act?
Potential Uses

Avoid TOCTTOU race
Potential Uses

Avoid TOCTTOU race

```c
int fd;
if (access(argv[1], R_OK) != 0)
    exit(1);
fd = open(argv[1], O_RDONLY);
```
Potential Uses

Avoid TOCTTOU race

```c
int fd;
if (access(argv[1], R_OK) != 0)
exit(1);
fd = open(argv[1], O_RDONLY);
```

- Can check exactly at time of use
Potential Uses

Guest profiling (think Systemtap)
Potential Uses

Process-specific tests
Potential Uses

Process-specific tests

• Custom to application
Potential Uses

Process-specific tests

- Custom to application
- User supplied?
Potential Uses

Process-specific tests
- Custom to application
- User supplied?
- Cloud provider perspective?
Potential Uses

Execution artifacts (signature)
Potential Uses

Execution artifacts (signature)

- Place probes in applications
Potential Uses

Execution artifacts (signature)

• Place probes in applications
• Characterize by execution pattern?
Potential Uses

Execution artifacts (signature)

- Place probes in applications
- Characterize by execution pattern?
- Invariants?
Performance
Hook/probe Latency

- Lares: 27 µs/hook (2.3 GHz CPU)
- hprobes: 3.2 µs/probe (3.0 GHz CPU)
- SIM: 0.4 µs/event (2.4 GHz CPU)

No tuning/perf modifications
Hook/probe Latency

- **Lares**: 27 $\mu$s/hook (2.3 GHz CPU)
- **hprobes**: 3.2 $\mu$s/probe (3.0 GHz CPU)
- **SIM**: 0.4$\mu$s/event (2.4 GHz CPU)
Hook/probe Latency

- Lares: $27 \, \mu s$/hook (2.3 GHz CPU)
- hprobes: $3.2 \, \mu s$/probe (3.0 GHz CPU)
- SIM: $0.4 \, \mu s$/event (2.4 GHz CPU)

No tuning/perf modifications
## Some quick benchmarks

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Run time (s) w/o probes*</th>
<th>Run time (s) w/ probes*</th>
<th>Percent Difference</th>
<th>Probe locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>find /</td>
<td>88.69 (0.27)</td>
<td>92.10 (1.98)</td>
<td>3.82%</td>
<td>sys_open</td>
</tr>
<tr>
<td>SPECjvm2008</td>
<td>360.5 (7.50)</td>
<td>492.43 (19.4)</td>
<td>36.6%</td>
<td>sys_open, sys_read, sys_write, sys_close</td>
</tr>
<tr>
<td>kernel make</td>
<td>245.15 (16.7)</td>
<td>250.06 (14.5)</td>
<td>2.00%</td>
<td>sys_open</td>
</tr>
<tr>
<td>kernel make</td>
<td>245.15 (16.7)</td>
<td>245.41 (16.3)</td>
<td>0.11%</td>
<td>sys_write</td>
</tr>
<tr>
<td>kernel make</td>
<td>245.15 (16.7)</td>
<td>245.13 (16.7)</td>
<td>0.01%</td>
<td>sys_read</td>
</tr>
<tr>
<td>kernel make</td>
<td>245.15 (16.7)</td>
<td>246.38 (16.7)</td>
<td>0.50%</td>
<td>sys_close</td>
</tr>
</tbody>
</table>
A Scaling Picture

Hprobe scaling with the number of probes

- median
- std deviation
- std err of μ

Run time (normalized to one probe)

Number of probes

Zak Estrada

Hprobes
Summary

- Hooks
- Lares, SIM, etc...
- How hprobes work
- Current status
- Potential uses
- Early performance data

Comments/questions/ideas?
Summary

- Hooks
- Lares, SIM, etc...
- How hprobes work
- Current status
- Potential uses
- Early performance data

Comments/questions/ideas?

Zak Estrada
\LaTeX \ text \ theme \ courtesy \ Flip \ Tanedo, \ Cornell

Images are rights of their respective owners