Bolt: I Know What You Did Last Summer... In the Cloud

Christina Delimitrou\textsuperscript{1} and Christos Kozyrakis\textsuperscript{2}

\textsuperscript{1}Cornell University, \textsuperscript{2}Stanford University

Platform Lab Review – February 2018
Executive Summary

- Problem: cloud resource sharing hides security vulnerabilities
  - Interference from co-scheduled apps \(\rightarrow\) leaks app characteristics

- Bolt: adversarial runtime in public clouds
  - Transparent & accurate app detection (5-10 sec)
    - User study: 88% correctly identified applications
  - Allows for difficult-to-detect DoS attacks
    - E.g. 140x increase in latency
  - Resource partitioning is helpful but insufficient
Motivation

App1  App2

Amazon Web Services
Windows Azure
Google Cloud Platform
Motivation

 cores

 App1

 App2

 Amazon Web Services

 Windows Azure

 Google Cloud Platform
Motivation

![Diagram showing cores and memory capacity for App1 and App2]

- **cores**
- **memory capacity**

- **App1**
- **App2**
Motivation

- Cores
- Memory capacity
- Storage capacity/bw
Motivation

app1

cores

memory capacity

network bw

app2

storage capacity/bw

AWS

Windows Azure

Google Cloud Platform
Motivation

- LL cache
- cores
- memory capacity
- network bw
- storage capacity/bw
- App1
- App2
Motivation

- LL cache
- cores
- memory capacity
- network bw
- storage capacity/bw
- power
Motivation

- Not all isolation techniques available
- Not all used/configured correctly
- Not all scale well
- Mem bw/core resources not isolated
Key idea: Leverage lack of isolation in public clouds to infer application characteristics
- Programming framework, algorithm, load characteristics

Exploit: enable practical, effective, and hard-to-detect performance attacks
- DoS, RFA, VM pinpointing
- Use app characteristics (sensitive resource) against it
- Avoid CPU saturation → hard to detect
Threat Model

- Active adversary but no control over VM placement
- Impartial, neutral cloud provider
1. Contention injection
2. Interference Impact measurement
3. App inference

Bolt

Adversary

Victim
Bolt

1. Contention injection
2. Interference impact measurement
3. App inference
4. Custom contention kernel
5. Performance attack

Adversary

Victim
1. Contention Measurement

- Set of contentious kernels (iBench)
  - Compute
  - L1/L2/L3
  - Memory bw
  - Storage bw
  - Network bw
  - (Memory/Storage capacity)

- Sample 2-3 kernels, run in adversarial VM

- Measure impact on performance of kernels vs. isolation
2. Practical App Inference

- Infer resource pressure in non-profiled resources
  - Sparse → dense information
  - SGD (Collaborative filtering)

- Classify unknown victim based on previously-seen applications
  - Label & determine resource sensitivity
  - Content-based recommendation

Hybrid recommender
1. **Infer pressure in non-profiled resources**
   - Reconstruct sparse information
   - Stochastic Gradient Descent (SGD), $O(mp_k)$

![Image showing the process of Big Data to the Rescue](image)
2. Classify and label victims

- Weighted Pearson Correlation Coefficients
- Output: distribution of similarity scores to app classes
Inference Accuracy

- **40 machine cluster (420 cores)**
- **Training apps:** 120 jobs (analytics, databases, webservers, in-memory caching, scientific, js) → high coverage of resource space
- **Testing apps:** 108 latency-critical webapps, analytics
- **No overlap in algorithms/datasets between training and testing sets**

<table>
<thead>
<tr>
<th>Application class</th>
<th>Detection accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-memory caching (memcached)</td>
<td>80%</td>
</tr>
<tr>
<td>Persistent databases (Cassandra, MongoDB)</td>
<td>89%</td>
</tr>
<tr>
<td>Hadoop jobs</td>
<td>92%</td>
</tr>
<tr>
<td>Spark jobs</td>
<td>86%</td>
</tr>
<tr>
<td>Webservers</td>
<td>91%</td>
</tr>
<tr>
<td><strong>Aggregate</strong></td>
<td><strong>89%</strong></td>
</tr>
</tbody>
</table>
3. Practical Performance Attacks

1. Determine the resource bottleneck of the victim
2. Create custom contentious kernel that targets critical resource(s)
3. Inject kernel in Bolt

- Several performance attacks (DoS, RFAs, VM pinpointing)
- Target specific, critical resource → low CPU pressure
3. Practical DoS Attacks

- Launched against same 108 applications as before
- On average 2.2x higher execution time and up to 9.8x
- For interactive services, on average 42x increase in tail latency and up to 140x

- Bolt does not saturate CPU
- Naïve attacker gets migrated
Demo

```
2. cd434@ath-1:~/matplotlib/bolt/bolt_demo$ ./adversary.sh

4. cd434@ath-1:~/matplotlib/bolt/bolt_demo$ ./victim2.sh
```
User Study

- 20 independent users from Stanford and Cornell

- Cluster
  - 200 EC2 servers, c3.8xlarge (32vCPUs, 60GB memory)

- Rules:
  - 4vCPUs per machine for Bolt
  - All users have equal priority
  - Users use thread pinning
  - Users can select specific instances

- Training set: 120 apps incl. analytics, webapps, scientific, etc.
Accuracy of App Labeling

Ground Truth

Correct app labels 63%

53 app classes (analytics, webapps, FS/OS, HLS/sim, other…)
Accuracy of App Characterization

Ground Truth
Correct app characteristics: 88%

Performance attack results in the paper
Is Isolation Enough?

- Need more scalable, fine-grain, and complete isolation techniques

![Bar Chart]

- None
- Thread Pinning
- +Net BW
- +Mem BW
- +Cache
- +Core Isolation

Accuracy (%)

- Baremetal
- Linux Containers
- Virtual Machines

- 45%
- 14%
Conclusions

- Bolt: highlight the security vulnerabilities from lack of isolation
  - Fast detection using online data mining techniques
  - Practical, hard-to-detect performance attacks
  - Current isolation helpful but insufficient

- In the paper:
  - Sensitivity to Bolt parameters
  - Sensitivity to applications and platform parameters
  - User study details
  - More performance attacks (resource freeing, VM pinpointing)