A Placement Vulnerability Study in Multi-tenant Public Clouds

Venkat(anathan) Varadarajan, Yinqian Zhang, Thomas Ristenpart and Michael Swift
venkatv@cs.wisc.edu
Public Clouds

Cloud API

Internet

Google Compute Engine

Microsoft Azure
Public Clouds

Cloud API

Internet

VM

VM

VM

VM

VM

VM

VM

VM

VM
Public Clouds

Cloud API

Internet

Multi-tenancy

VM

VM

VM

VM

VM

VM

VM

VM

Public Clouds

Multi-tenancy
Public Clouds

Cloud API

Internet

Multi-tenancy + Public

VM
VM
VM
VM
VM
VM
VM
Public Clouds

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location
Public Clouds

Cloud API

Internet

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location
Public Clouds

Cloud API

Internet

Multi-tenancy + Public

Big concern: cross-VM attacks via co-location
Public Clouds

We show attackers can achieve co-location with > 90% chance for as low as 14 cents!

Big concern: cross-VM attacks via co-location
Placement and Isolation in Public Clouds
Placement and Isolation in Public Clouds

Cloud API

Internet

Cluster Scheduler

Placement Policy

[Diagram of cloud servers connected via Cloud API and Placement Policy]

3
Placement and Isolation in Public Clouds

Cluster Scheduler

Cloud API

Placement Policy

Internet

VM

VM

VM

VM

VM

VM

VM
Placement and Isolation in Public Clouds

Cloud API

Internet

Cluster Scheduler

Placement Policy

(1) Controls co-location

Hypervisor
Placement and Isolation in Public Clouds

(1) Controls co-location

(2) Provides isolation
Placement and Isolation in Public Clouds

Cloud API

Internet

Cluster Scheduler

Placement Policy

(1) Controls co-location

(2) Provides isolation

- Cross-VM side- and covert-channels
  [Ristenpart’09, Xu’11, Wu’12, Zhang’12, Yarom’14, Liu’15, Gruss’15]

- Resource-stealing, DoS
  [Zhou’11, Varadarajan’12]
Placement and Isolation in Public Clouds

Cloud API

Internet

Cluster Scheduler

Placement Policy

Hypervisor

(1) Controls co-location

(2) Provides isolation

• Cross-VM side- and covert-channels [Ristenpart’09, Xu’11, Wu’12, Zhang’12, Yarom’14, Liu’15, Gruss’15]

• Resource-stealing, DoS [Zhou’11, Varadarajan’12]
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Prior Work on Co-location: 2009 Study

*Hey you get off my cloud, Ristenpart et al., CCS 2009
Prior Work on Co-location: 2009 Study

- 6 years old

*Hey you get off my cloud, Ristenpart et al., CCS 2009
Prior Work on Co-location: 2009 Study

- 6 years old
- New countermeasures (e.g., virtual private clouds)

*Hey you get off my cloud, Ristenpart et al., CCS 2009
Prior Work on Co-location: 2009 Study

- 6 years old
- New countermeasures (e.g., virtual private clouds)
- Increased scale of clouds

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Prior Work on Co-location: 2009 Study

- 6 years old
- New countermeasures (e.g., virtual private clouds)
- Increased scale of clouds
- Only on Amazon EC2

*Hey you get off my cloud, Ristenpart et al., CCS 2009*
Our Work: Exploring Co-location Attacks in Modern Clouds
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

Cloud API

Cluster Scheduler

Placement Policy

Target VMs
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

Cloud API

Internet

Cluster Scheduler

Placement Policy

Target VMs
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location

2. Detecting Co-location
   - with any target victim
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location

2. Detecting Co-location
   - with any target victim
Co-residency Detection
Co-residency Detection

1. Read shared state on two VMs
   e.g., private IP addresses, shared TSC counters.

102.2.1.1  102.2.1.3
Co-residency Detection

1. Read shared state on two VMs
   e.g., private IP addresses, shared TSC counters.

   102.2.1.1  102.2.1.3

2. Correlate performance of shared resources
   e.g., network round-trip times, cache-based covert-channels.

   n/w pings or covert-channels
Co-residency Detection

1. Read shared state on two VMs
e.g., private IP addresses, shared TSC counters.

2. Correlate performance of shared resources
e.g., network round-trip times, cache-based covert-channels.
Co-residency Detection

1. Read shared state on two VMs
e.g., private IP addresses, shared TSC counters.

2. Correlate performance of shared resources
e.g., network round-trip times, cache-based covert-channels.

A memory-based covert-channel* can cause 3x-4x degradation

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:

```c
Signal() {
  repeat
    atomic_op(ua_addr)
  done
}
```

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:

Signal() {
    repeat
        atomic_op(ua_addr)
    done
}

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:
Signal() {
    repeat
        atomic_op(ua_addr)
    done
}

Receiver:
Observe() {
    s = start_time
    repeat N
        mem_access()
    done
    e = end_time
    bw = N/(e-s)
}

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:
Signal() {
    repeat
        atomic_op(ua_addr)
    done
}

Receiver:
Observe() {
    s = start_time
    repeat N
        mem_access()
    done
    e = end_time
    bw = N/(e-s)
}

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:
Signal() {
    repeat
        atomic_op(ua_addr)
    done
}

Receiver:
Observe() {
    s = start_time
    repeat N
        mem_access()
    done
    e = end_time
    bw = N/(e-s)
}

“Victim” VM must cooperate with attack VM
O.K. for measurement studies

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Background: Memory Covert-Channel

Sender:

```c
Signal() {
    repeat
        atomic_op(ua_addr)
    done
}
```

Receiver:

```c
Observe() {
    s = start_time
    repeat N
        mem_access()
    done
    e = end_time
    bw = N/(e-s)
}
```

“Victim” VM must cooperate with attack VM
O.K. for measurement studies
But not useful for real attacks in the wild

* Wu et al. “Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.”, Usenix Security’12
Co-residency Detection on Uncooperative Victim
Co-residency Detection on Uncooperative Victim
Co-residency Detection on Uncooperative Victim

Target Victims

part of a cloud app, e.g., web server, key-value stores
Co-residency Detection on Uncooperative Victim

Target Victims

part of a cloud app, e.g., web server, key-value stores

Data Store
Co-residency Detection on Uncooperative Victim

Target Victims

Load Balancer

VM

VM

VM

Data Store

part of a cloud app, e.g., web server, key-value stores
Co-residency Detection on Uncooperative Victim

Target Victims

part of a cloud app, e.g., web server, key-value stores
Co-residency Detection on Uncooperative Victim

Target Victims

part of a cloud app, e.g., web server, key-value stores

Load Balancer

VM

VM

VM

Data Store
Co-residency Detection on Uncooperative Victim

Target Victims

part of a cloud app, e.g., web server, key-value stores

Load Balancer
Co-residency Detection on Uncooperative Victim

Realistic victim setting: modern multi-tier cloud app.
Co-residency Detection on Uncooperative Victim

• Realistic victim setting: modern multi-tier cloud app.

part of a cloud app, e.g., web server, key-value stores
Co-residency Detection on Uncooperative Victim

- Realistic victim setting: modern multi-tier cloud app.
Co-residency Detection on Uncooperative Victim

- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic
Co-residency Detection on Uncooperative Victim

- Realistic victim setting: modern multi-tier cloud app.
- A way to detect co-location when VMs are behind a load-balancer in presence of background traffic.
Co-residency Detection on Uncooperative Victim

• Realistic victim setting: modern multi-tier cloud app.
• A way to detect co-location when VMs are behind a load-balancer in presence of background traffic
Co-residency Detection on Uncooperative Victim

Realistic victim setting: modern multi-tier cloud app.

A way to detect co-location when VMs are behind a load-balancer in presence of background traffic.
Co-residency Detection on Uncooperative Victim

Social networking application (Olio): HAProxy LB + 3 Web servers + 1 mysql server
Co-residency Detection on Uncooperative Victim

Social networking application (Olio): HAProxy LB + 3 Web servers + 1 mysql server
Co-residency Detection on Uncooperative Victim

Social networking application (Olio): HAProxy LB + 3 Web servers + 1 mysql server

Average Request Latency (ms) (log scale)

Background Load (# concurrent users)

idle 100 250 500 750 1000

Non-coresident  Co-resident
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location

2. Detecting Co-location
   - with any target victim
Our Work: Exploring Co-location Attacks in Modern Clouds

Steps in achieving co-location:

1. Finding Launch Strategy
   - launch parameters to increase chances of co-location

2. Detecting Co-location
   - with any target victim
Big Picture: Placement Vulnerability Study
Big Picture: Placement Vulnerability Study
Big Picture: Placement Vulnerability Study

e.g., # VMs, when you launch, datacenter, VM type, etc.
Big Picture: Placement Vulnerability Study

e.g., # VMs, when you launch, datacenter, VM type, etc.

Placement Policy

Co-location?

Placement Variables
Big Picture: Placement Vulnerability Study

Placement Variables

Co-location?

e.g., # VMs, when you launch, datacenter, VM type, etc.

Fix Placement Variables
Big Picture: Placement Vulnerability Study

e.g., # VMs, when you launch, datacenter, VM type, etc.

Placement Variables

Fix Placement Variables

Observe Placement Behavior

Placement Policy

Co-location?
Big Picture: Placement Vulnerability Study

Placement Variables
Co-location?

Fix Placement Variables
Observe Placement Behavior

Placement Policy

e.g., # VMs, when you launch, datacenter, VM type, etc.
Big Picture: Placement Vulnerability Study

Study spanning 3 months, exploring 6 placement variables, spending more than $200 per cloud

e.g., # VMs, when you launch, datacenter, VM type, etc.
Big Picture: Placement Vulnerability Study

Study spanning 3 months, exploring 6 placement variables, spending more than $200 per cloud

e.g., # VMs, when you launch, datacenter, VM type, etc.
Study Setup

- Two distinct accounts: proxy for victim and attacker
Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
  - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
  - Small instance type (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
  - Values for these variables form a launch strategy
Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
  - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
  - Small instance type (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
  - Values for these variables form a launch strategy
- Execute a launch strategy from a workstation
  - detect and log co-location
Study Setup

- Two distinct accounts: proxy for victim and attacker
- 6 placement variables
  - # victim & attacker VMs, delay b/w launches, time of day, day of week, datacenter, cloud providers
  - Small instance type (EC2: t2.small, GCE: g1.small, Azure: Standard-A1)
  - Values for these variables form a launch strategy
- Execute a launch strategy from a workstation
  - detect and log co-location
- 9 samples per strategy with 3 runs per time of day & 2 days of week (weekday/weekend)
How hard should it be to achieve co-location?
How hard should it be to achieve co-location?

- Random placement policy
How hard should it be to achieve co-location?

- Random placement policy
- $N = 50,000$ machines [re:Invent’14]
How hard should it be to achieve co-location?

- Random placement policy
- $N = 50,000$ machines [re:Invent’14]
- $v$ - victims and $a$ - attacker VMs
How hard should it be to achieve co-location?

- Random placement policy
- \( N = 50,000 \) machines \([\text{re:Invent'14}]\)
- \( v \) - victims and \( a \) - attacker VMs
- Probability of Collision:
  \[ P_c = 1 - (1 - v/N)^a \]
How hard should it be to achieve co-location?

- Random placement policy
- $N = 50,000$ machines $\text{[re:Invent'14]}$
- $v$ - victims and $a$ - attacker VMs
- Probability of Collision:
  $P_c = 1 - (1 - v/N)^a$

\[ v = \ln(1 - P_c)/\ln(1 - v/N); \quad P_c = 0.5 \]
How hard should it be to achieve co-location?

- Random placement policy
- $N = 50,000$ machines [re:Invent’14]
- $v$ - victims and $a$ - attacker VMs
- Probability of Collision:
  \[ P_c = 1 - (1 - \frac{v}{N})^a \]

<table>
<thead>
<tr>
<th>$v$</th>
<th>$a = \ln(1 - P_c)/\ln(1 - \frac{v}{N})$; $P_c = 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3466</td>
</tr>
<tr>
<td>20</td>
<td>1733</td>
</tr>
<tr>
<td>30</td>
<td>1155</td>
</tr>
</tbody>
</table>
How hard should it be to achieve co-location?

- Random placement policy
- \( N = 50,000 \) machines [re:Invent’14]
- \( v \) - victims and \( a \) - attacker VMs
- Probability of Collision:
  \[ P_c = 1 - (1 - \frac{v}{N})^a \]

For a modest 50% success rate with 10-30 victims we need to launch \(~3000\) VMs

<table>
<thead>
<tr>
<th>( v )</th>
<th>( a = \ln(1 - P_c)/\ln(1 - v/N) ); ( P_c = 0.5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3466</td>
</tr>
</tbody>
</table>
## Results: Varying Number of VMs

<table>
<thead>
<tr>
<th>Success Rate</th>
<th>EC2</th>
<th>GCE</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

vm types: 10v, 20v, 30v, 10a, 20a, 30a
Results: Varying Number of VMs

<table>
<thead>
<tr>
<th>Success Rate</th>
<th>EC2</th>
<th>GCE</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10v 10a</td>
<td>10v 10a</td>
<td>10v 10a</td>
</tr>
<tr>
<td>0.25</td>
<td>10v 20a</td>
<td>10v 20a</td>
<td>10v 20a</td>
</tr>
<tr>
<td>0.5</td>
<td>10v 30a</td>
<td>20v 30a</td>
<td>20v 30a</td>
</tr>
<tr>
<td>0.75</td>
<td>30v 30a</td>
<td>30v 30a</td>
<td>30v 30a</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>10v 10a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>10v 10a</th>
<th>10v 20a</th>
<th>10v 30a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results: Varying Number of VMs

- **EC2**
  - Success Rate:
    - 0
    - 0.25
    - 0.5
    - 0.75
    - 1
  - VM Configurations:
    - 10v 10a
    - 10v 20a
    - 10v 30a
    - 20v 30a
    - 30v 30a

- **GCE**
  - Success Rate:
    - 0
    - 0.25
    - 0.5
    - 0.75
    - 1
  - VM Configurations:
    - 10v 10a
    - 10v 20a
    - 10v 30a
    - 20v 30a
    - 30v 30a

- **Azure**
  - Success Rate:
    - 0
    - 0.25
    - 0.5
    - 0.75
    - 1
  - VM Configurations:
    - 10v 10a
    - 10v 20a
    - 10v 30a
    - 20v 30a
    - 30v 30a
## Results: Varying Number of VMs

<table>
<thead>
<tr>
<th>Platform</th>
<th>Success Rate</th>
<th>VM Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2</td>
<td>0.25</td>
<td>10v 10a</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>10v 20a</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>10v 30a</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>30v 30a</td>
</tr>
<tr>
<td>GCE</td>
<td>0</td>
<td>10v 10a</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>10v 20a</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>10v 30a</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>20v 30a</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>30v 30a</td>
</tr>
<tr>
<td>Azure</td>
<td>0.25</td>
<td>10v 10a</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>10v 20a</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>10v 30a</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>20v 30a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30v 30a</td>
</tr>
</tbody>
</table>

Co-location is possible with as low as 10 VMs and always achieve co-location with 30 VMs.
## Results: Varying Delay between Launches

<table>
<thead>
<tr>
<th>Success Rate</th>
<th>EC2</th>
<th>GCE</th>
<th>Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.25</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.75</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Zero* vs *One hour*
Results: Varying Delay between Launches

- **EC2**
  - Success Rate
  - Zero and One hour

- **GCE**
  - Success Rate
  - Zero and One hour

- **Azure**
  - Success Rate
  - Zero and One hour
Results: Varying Delay between Launches

- **EC2**
  - Success Rate
    - 10v 10a
    - 20v 20a
    - 30v 30a

- **GCE**
  - Success Rate
    - 10v 10a
    - 20v 20a
    - 30v 30a

- **Azure**
  - Success Rate
    - 10v 10a
    - 20v 20a
    - 30v 30a
Results: Varying Delay between Launches

EC2

Success Rate

GCE

Zero
One hour

Azure

0
10v 10a
20v 20a
30v 30a
Results: Varying Delay between Launches

Placement policy for each cloud significantly varies.
Cost of a Launch Strategy

Time to achieve & detect co-location (in minutes)

- 0 minutes: $0.00
- 45 minutes: $1.50
- 90 minutes: $3.00
- 135 minutes: $4.50
- 180 minutes: $6.00

Cost

- EC2
- GCE
- AZURE

10v 10a
20v 20a
30v 30a
Cost of a Launch Strategy

<table>
<thead>
<tr>
<th>Time to achieve &amp; detect co-location (in minutes)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>10v 10a</td>
<td>$0.00</td>
</tr>
<tr>
<td>20v 20a</td>
<td>$1.50</td>
</tr>
<tr>
<td>30v 30a</td>
<td>$3.00</td>
</tr>
</tbody>
</table>

Platforms:
- EC2
- GCE
- AZURE
Cost of a Launch Strategy

<table>
<thead>
<tr>
<th>Time to achieve &amp; detect co-location (in minutes)</th>
<th>EC2</th>
<th>GCE</th>
<th>AZURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10v 10a</td>
<td>$0.00</td>
<td>$1.50</td>
<td>$3.00</td>
</tr>
<tr>
<td>20v 20a</td>
<td>$1.50</td>
<td>$3.00</td>
<td>$4.50</td>
</tr>
<tr>
<td>30v 30a</td>
<td>$3.00</td>
<td>$4.50</td>
<td>$6.00</td>
</tr>
</tbody>
</table>
The cheapest launch strategy costs as low as 14 cents
Other Interesting Results
Other Interesting Results

- We always achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
Other Interesting Results

- We always achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
  - In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
  - In Azure we could co-locate 16 VMs on a single host
Other Interesting Results

- We *always* achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
- In Azure we could co-locate 16 VMs on a single host
- Brief experiments with platform-as-a-service,
Other Interesting Results

- We always achieved co-location in smaller datacenter regions,
  - GCE: europe-west1-b and EC2: us-west-1 (CA)
- In EC2, launching attacker VMs early morning (2 to 10am PST) has a higher success rate.
- In Azure we could co-locate 16 VMs on a single host
- Brief experiments with platform-as-a-service,
- … and many more in the paper
Some Strategies Work Better than Others
# Some Strategies Work Better than Others

## Example strategies on EC2

<table>
<thead>
<tr>
<th>Launch Strategy</th>
<th>v x a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch 10 VMs in less popular datacenter</td>
<td>10x10</td>
</tr>
<tr>
<td>Launch 30 VMs 1 hour after victim VM launches</td>
<td>30x30</td>
</tr>
<tr>
<td>Launch more than 20 VMs 4 hours after victim VM launches</td>
<td>20x20</td>
</tr>
</tbody>
</table>
### Some Strategies Work Better than Others

#### Example strategies on EC2

<table>
<thead>
<tr>
<th>Launch Strategy</th>
<th>v x a</th>
<th>Cost in Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch 10 VMs in less popular datacenter</td>
<td>10x10</td>
<td>$0.26</td>
</tr>
<tr>
<td>Launch 30 VMs 1 hour after victim VM launches</td>
<td>30x30</td>
<td>$1.56</td>
</tr>
<tr>
<td>Launch more than 20 VMs 4 hours after victim VM launches</td>
<td>20x20</td>
<td>$0.52</td>
</tr>
</tbody>
</table>
### Some Strategies Work Better than Others

#### Example strategies on EC2

<table>
<thead>
<tr>
<th>Launch Strategy</th>
<th>v x a</th>
<th>Cost in Cloud</th>
<th>Cost under Random Placement*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch 10 VMs in less popular datacenter</td>
<td>10x10</td>
<td>$0.26</td>
<td>$113.87</td>
</tr>
<tr>
<td>Launch 30 VMs 1 hour after victim VM launches</td>
<td>30x30</td>
<td>$1.56</td>
<td>$32.75</td>
</tr>
<tr>
<td>Launch more than 20 VMs 4 hours after victim VM launches</td>
<td>20x20</td>
<td>$0.52</td>
<td>$53.76</td>
</tr>
</tbody>
</table>

*v x a launch strategy has a probability of collision: 1 - (1 - v/N)^a*
Some Strategies Work Better than Others

Example strategies on EC2

<table>
<thead>
<tr>
<th>Launch Strategy</th>
<th>v x a</th>
<th>Cost in Cloud</th>
<th>Cost under Random Placement*</th>
<th>Success rate norm. w/ random*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch 10 VMs in less popular datacenter</td>
<td>10x10</td>
<td>$0.26</td>
<td>$113.87</td>
<td>1/0.1 (=10)</td>
</tr>
<tr>
<td>Launch 30 VMs 1 hour after victim VM launches</td>
<td>30x30</td>
<td>$1.56</td>
<td>$32.75</td>
<td>1/0.6 (=1.67)</td>
</tr>
<tr>
<td>Launch more than 20 VMs 4 hours after victim VM launches</td>
<td>20x20</td>
<td>$0.52</td>
<td>$53.76</td>
<td>1/0.33 (=3.03)</td>
</tr>
</tbody>
</table>

*Random Placement of VMs on N hosts, v x a launch strategy has a probability of collision: 1 - (1 - v/N)^a
Summary: Co-location Attacks in Modern Clouds

- (1) Controls co-location
- (2) Provides isolation
  - Cross-VM Side- and covert-channels
    [Ristenpart’09, Xu’11, Wu’12, Zhang’12, Yarom’14, Liu’14, Gruss’15]
  - Resource-stealing, DoS
    [Zhou’11, Varadarajan’12]

Co-location attacks are **practical** and **very cheap** in modern clouds!