Project Silver

Rethinking Security in the Era of Cloud Computing

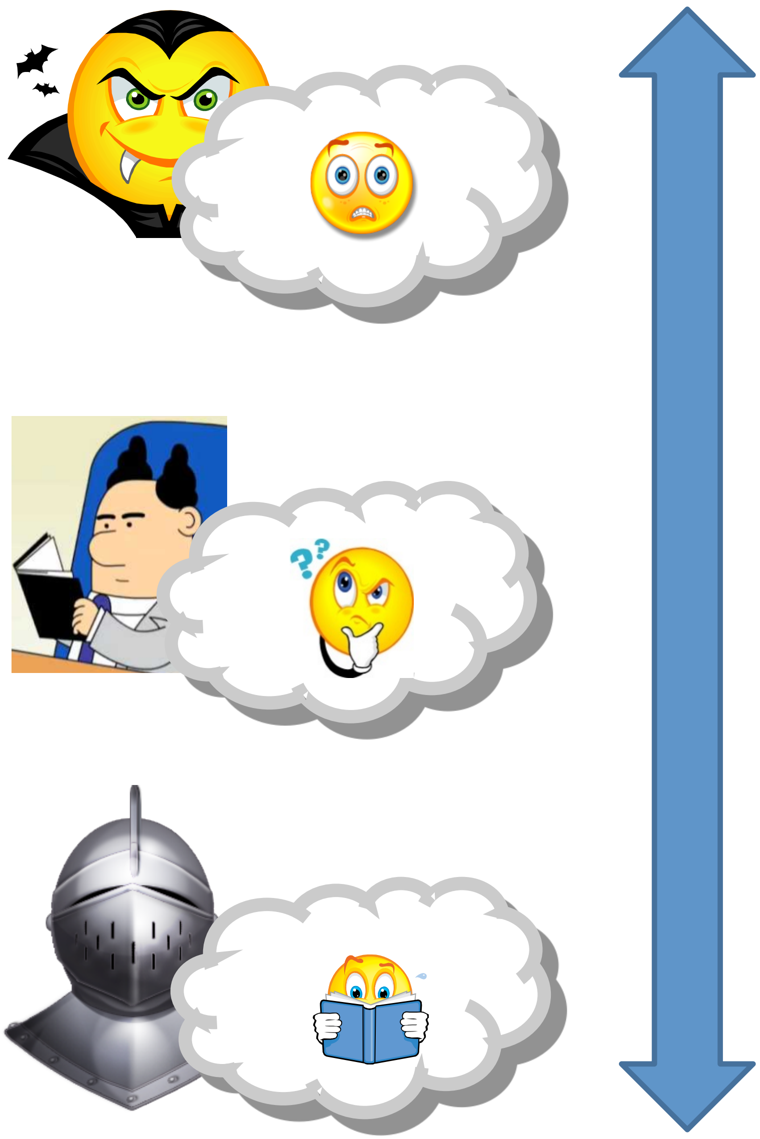
Jay Aikat, Aditya Akella, Jeff Chase,
Will Enck, Ari Juels, **Michael K. Reiter**,
Thomas Ristenpart, Vyas Sekar, Michael Swift

The University of North Carolina at Chapel Hill
Carnegie Mellon University
Cornell Tech
Duke University
NC State University
Wisconsin University of Wisconsin–Madison

The Move to Cloud Computing

- >7% of the Alexa top 1M websites are tenants on EC2 or Azure
- Technical trends
  - Centralization in big providers
  - Clouds with more features

Threat Models



- The cloud *is* the adversary

  ⟹ e.g., virtualization secure against hypervisor, fully homomorphic encryption

- The cloud needs help

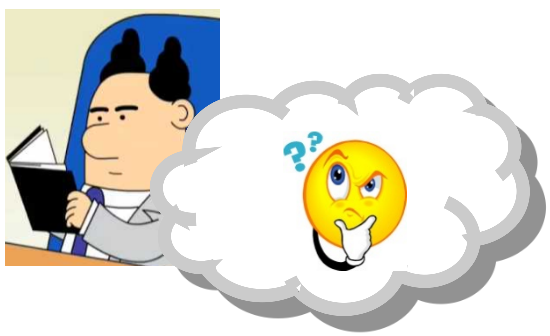
  ⟹ e.g., cycle stealing, colocation, cartography, side channels

- The cloud is an asset

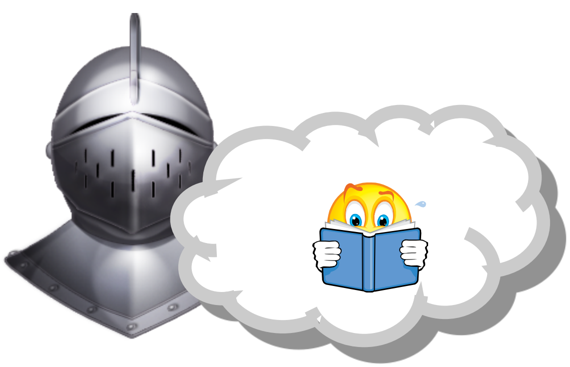
  ⟹ can be leveraged to do things that we couldn't do before

Reconsidering the Threat Model

"Most" academic research today is here …



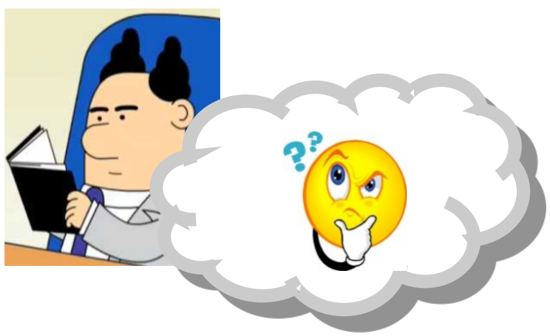
We want to be here …

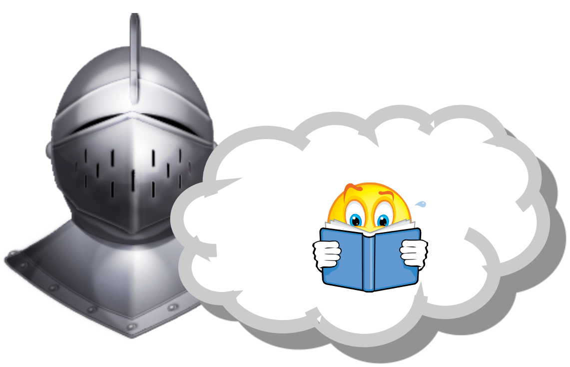


… and especially here.

Reconsidering the Threat Model

At odds with industry realities and incentives





- Better aligned with industry
- Easier deployment paths
- An understudied opportunity

The Driving Vision

- A "cloud control platform" that supports
  - Improved cloud and tenant security
  - Innovative services to enable new modes of tenant interaction

- … through new tech for better managing
  - Tenants' clients (credentials, protocols, …)
  - Tenant infrastructure (outsourced services, …)
  - Tenant-to-tenant ecosystem (trust management)

Cloud Security Horizons (CSH) Summits

- Three Cloud Security Horizon "summits"

- First CSH held in Feb 2014 in San Francisco
  - Co-located with the RSA Conference
- Second CSH held in Mar 2016 in New York City
- Last CSH Summit to be held in Spring 2018
  - Location TBD

Motivation for CSH

- Summits where we gather with industry stakeholders for technical exchange
  - Talks from both research team and industry
  - Facilitate technology flow and knowledge exchange
  - Focus discussions around the realities of cloud computing security
  - Familiarize industry partners with our tools and research directions
  - Industry partners serve an informal advisory role for our project

Cloud Security Curriculum Development Workshop

- 3-day curriculum workshop to help college teachers integrate cloud security into their courses

- Goal: curricular materials with integrated cloud security components …
  - From different perspectives
  - From different institutions
  - Within diverse courses

Cloud Security Curriculum Development Workshop

- First CSCD workshop held Jul 15-17, 2015 in Chapel Hill, NC
- Second CSCD workshop held Jul 13-15, 2016, also in Chapel Hill, NC

Cloud Security Curriculum Development Workshop

Day One (Wednesday, July 13)

- 08:30 – 09:00 Breakfast and Registration
- 09:00 – 09:40 Welcome, introductions, final agenda
- 09:40 – 10:00 Introduction to Cloud Computing and Cloud Security
- 10:00 – 10:45 Cloud 101 project – hands-on tutorial using **Amazon EC2**
- 11:00 – 12:00 Presentation of the Silver CSCW modules and their potential usage in classes (with examples for Distributed Systems, Introduction to Security, and Networking courses)
- 12:00 – 13:00 Lunch
- 13:00 – 15:00 **Cloud Security using GENI: demo and hands-on tutorial**
- 15:15 – 16:15 **GENI tutorial on OpenFlow and NAT devices** (continued)…
- 16:15 – 16:30 Agenda for tomorrow; and Q&A

Cloud Security Curriculum Development Workshop

Day Two (Thursday, July 14)

- 08:30 – 09:00 Breakfast
- 09:00 – 10:30 **CloudLab: demo and hands-on tutorial**
- 10:45 – 11:15 CloudLab tutorial (contd…)
- 11:00 – 12:00 Gary Bishop: My experience with Docker
- 12:00 – 14:00 Lunch (en route to IBM Data Center); Travel by pre-arranged vans
- 14:00 – 16:00 **IBM Data Center tour**
- 18:00 – 20:00 Working Dinner: Breakout sessions – pick your module and plan the implementation in your course(s)

Cloud Security Curriculum Development Workshop

Day Three (Friday, July 15)

- 09:00 – 09:30 Breakfast
- 09:30 – 09:45 Talk about a course experience by one of the participants
- 09:45 – 10:15 Mike Reiter – Side-channel attacks
- 10:15 – 10:30 Introduction to other Educational Resources
- 10:45 – 12:00 5 to 6-min presentations by each participant on how they plan to use our modules
- 12:00 – 13:00 Lunch – wrap-up, feedback, and next steps.

The Driving Vision

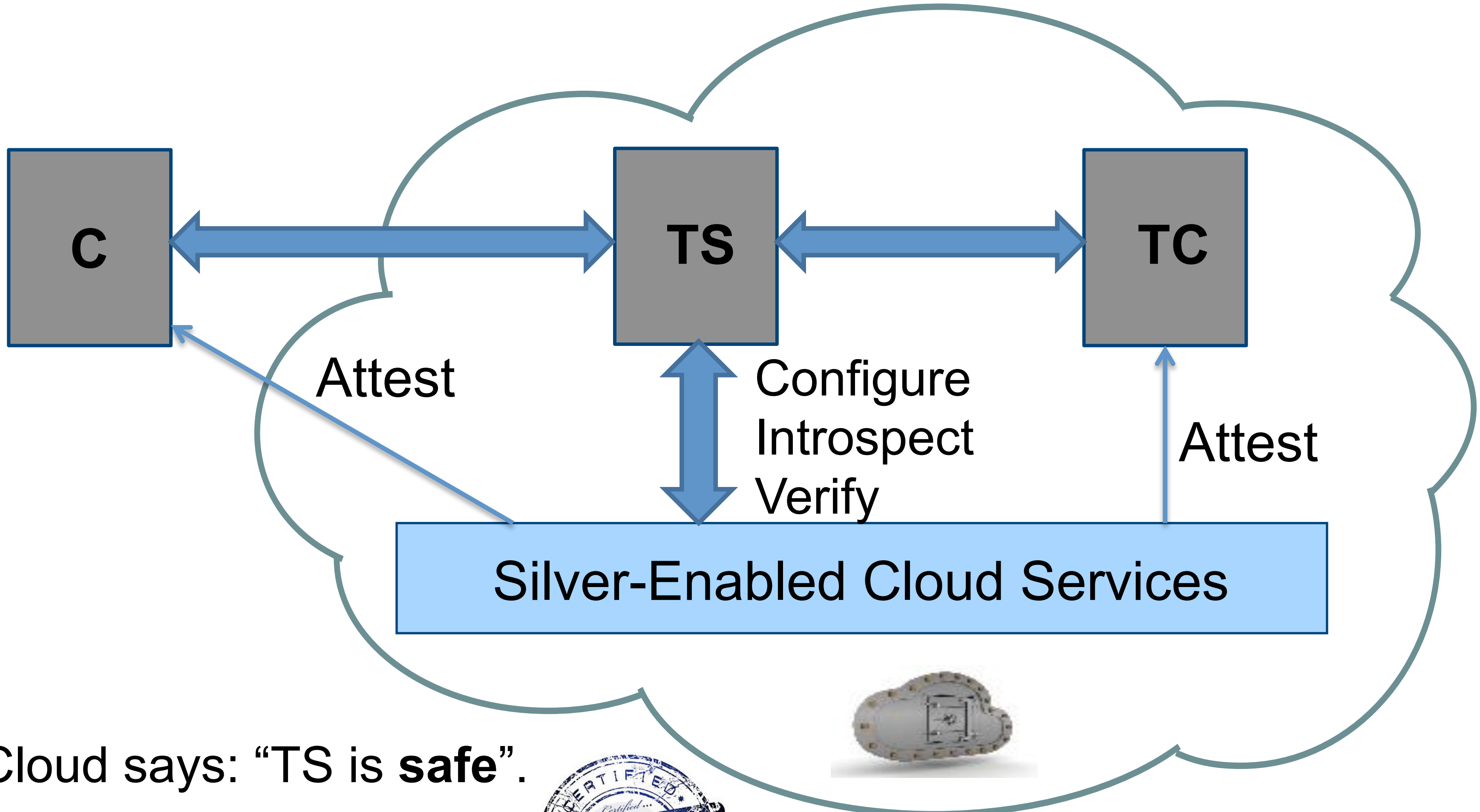
- A "cloud control platform" that supports
  - Improved cloud and tenant security
  - Innovative services to enable new modes of tenant interaction

- … through new tech for better managing
  - Tenants' clients (credentials, protocols, …)
  - Tenant infrastructure (outsourced services, …)
  - Tenant-to-tenant ecosystem (trust management)

Strengthening Tenant Ecosystems

- **Focus**: New provider services to certify/attest tenant configurations and security properties.
  - **Leverage trust in cloud provider**
  - Broker trust among tenants
  - Evidence for regulatory/policy compliance
  - Practical code attestation → trusted instances
  - Extend authz for attribute-based access
  - Make trust relationships explicit
  - **Speculative**: requires new trust framework

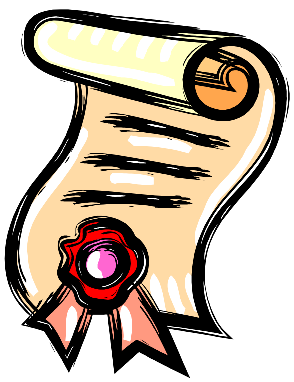
Attesting Security Properties



Cloud says: “TS is **safe**”.

Examples (Vision)

"TS is running ***SELinux*** version X.Y.Z, ***fully patched***"

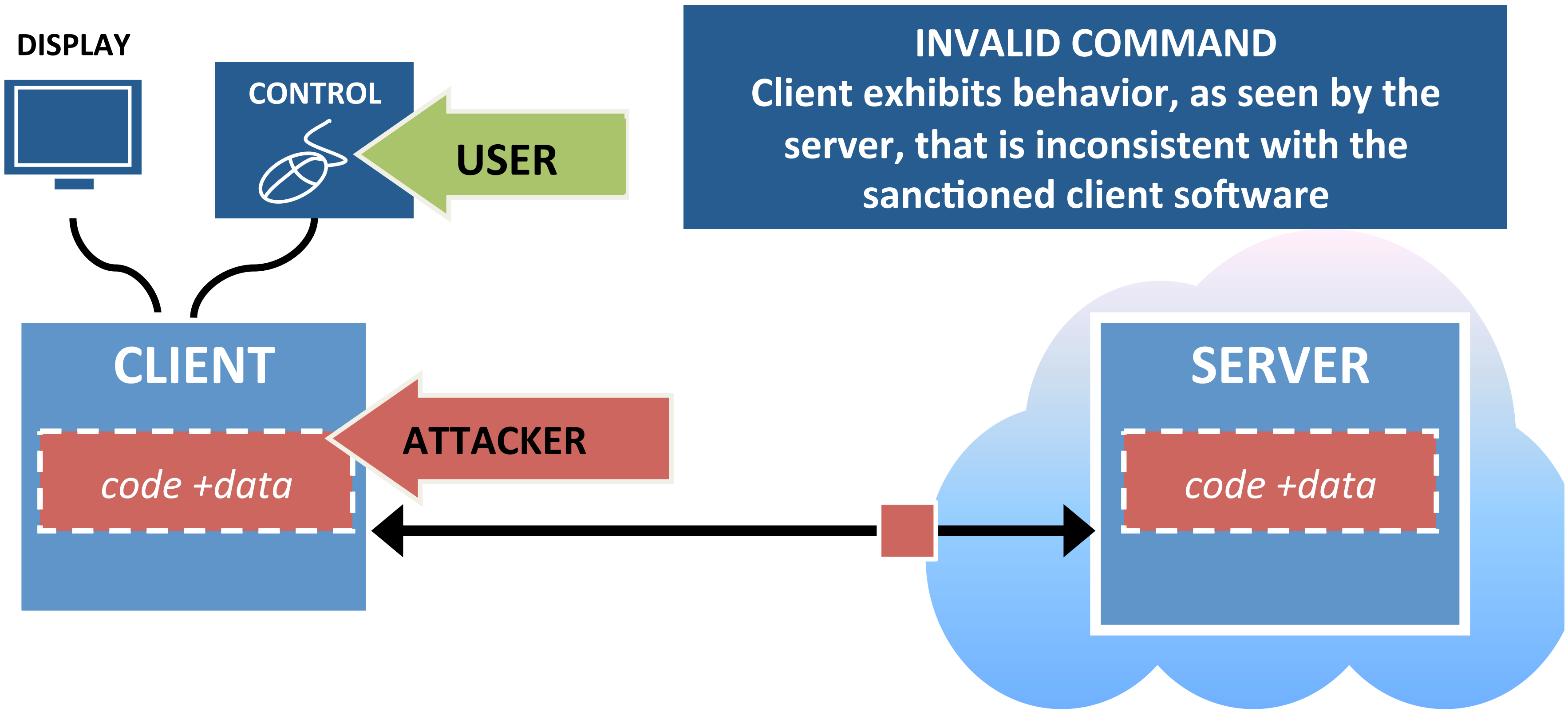


Proof of: "TS's security posture is ISO ***XYZ-compliant***"

"TS ***cannot leak data*** except via the approved output channel."

"TS is a ***sealed, immutable instance*** of application XYZ."

Invalid Command Attacks



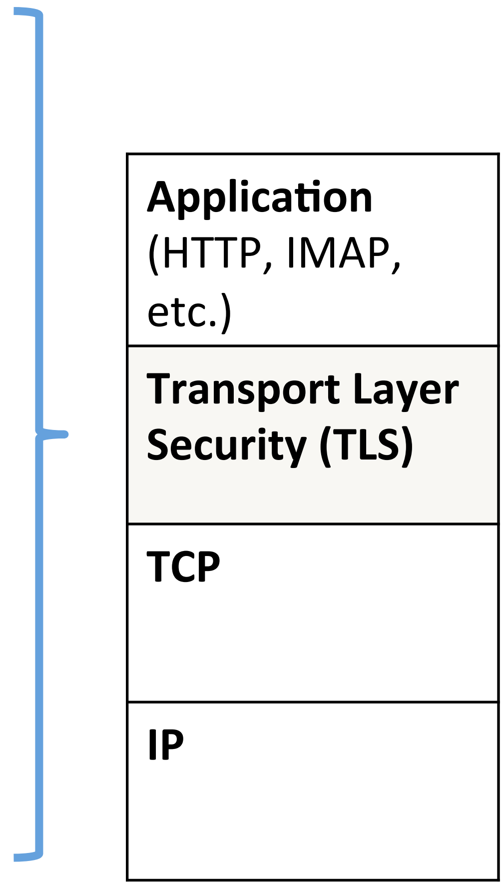
Invalid Command Attacks

- Tampering with clients in client-server protocols is an ingredient in numerous abuses
  - Exploits on the server directly
  - Manipulation of client state for which it is authoritative

- Exploits can take the form of …
  - Cleverly crafted malicious packets, or
  - Sequences of individually valid packets that exploit flaws in server logic or limitations in server visibility

Transport Layer Security (TLS)

- Handshake Protocol
  - Select cipher, authentication, key exchange
- Heartbeat Protocol
- Record Layer
  - Provides confidentiality and integrity
  - Encapsulates other protocols (above)

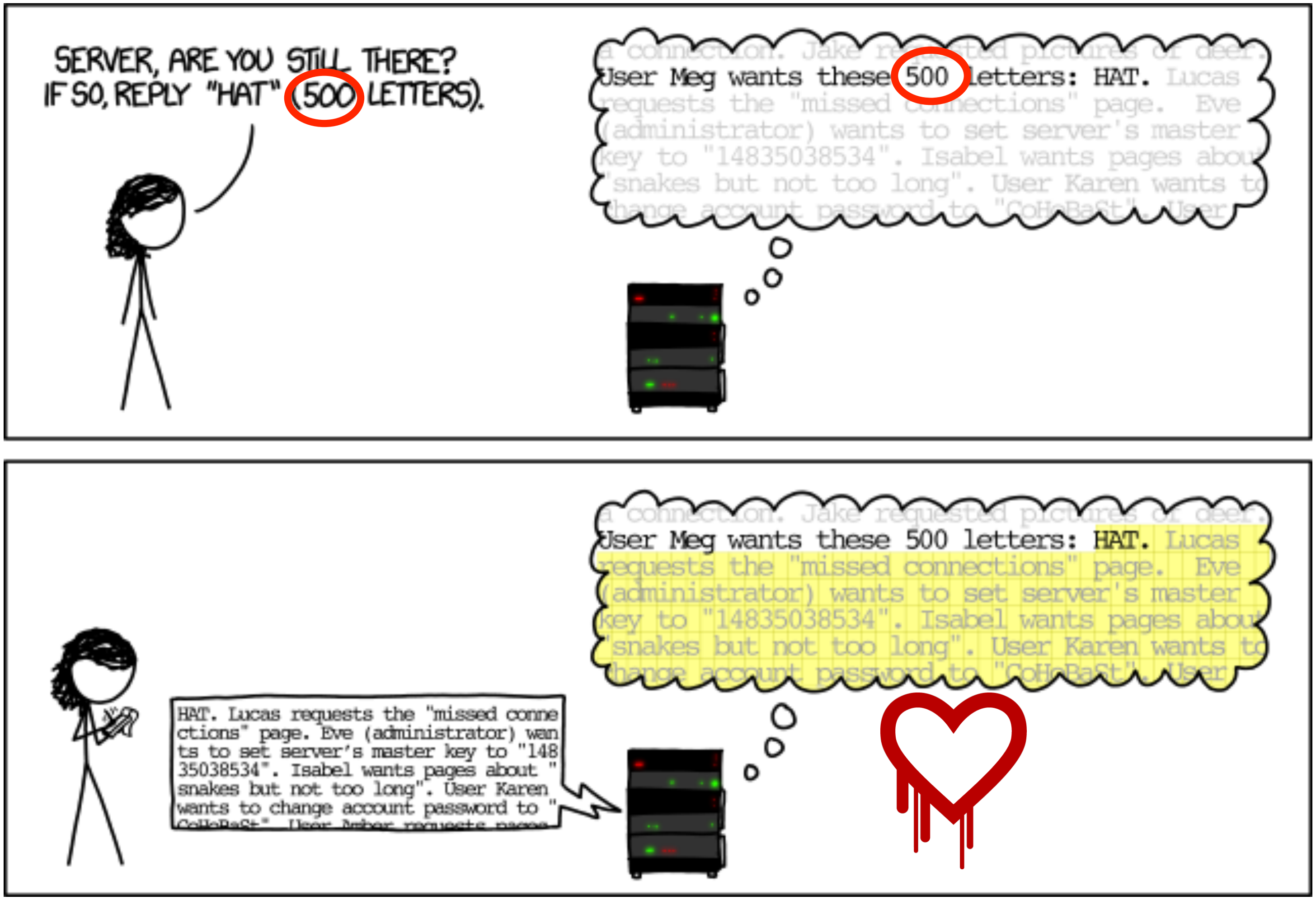


In 2014, critical vulnerabilities were discovered in all 5 major implementations of TLS (including OpenSSL).

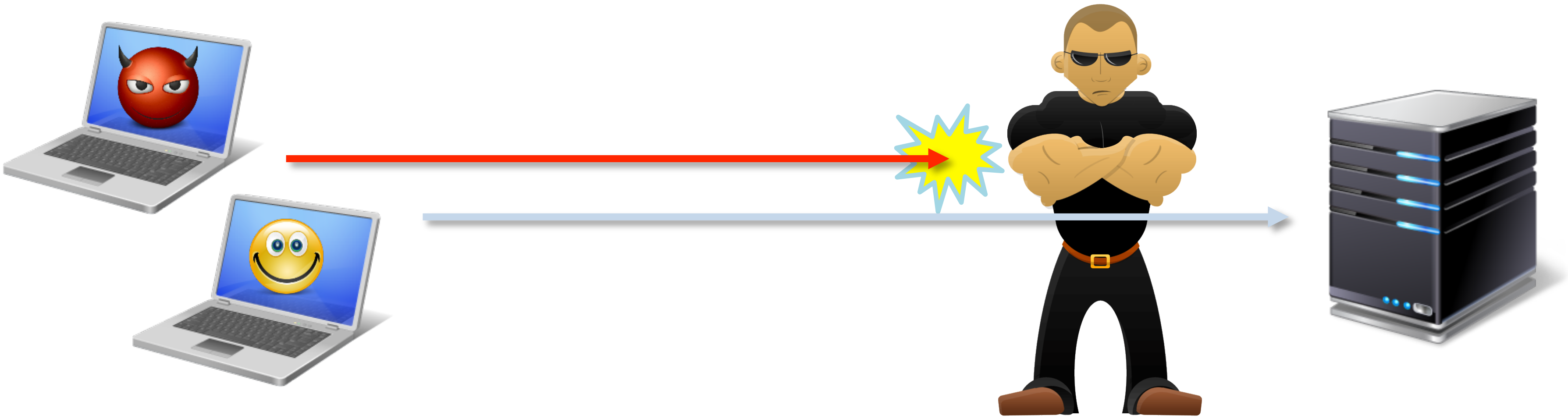
Heartbleed

- Implementation bug in OpenSSL (TLS Heartbeat handler)
- Nearly all OpenSSL applications vulnerable for 2 years
- 17% of the Internet's web servers (~500,000)
- Not just web: IMAP/SMTP, VPN, Android 4.1.1, etc.
- 4 months later, half remained unpatched (IBM, 3Q 2014)
- Even worse, patching is insufficient
  - Certificates must be revoked and reissued
  - Only 13% of vulnerable websites did so (Zhang et al., 2014)

Heartbleed (CVE-2014-0160)

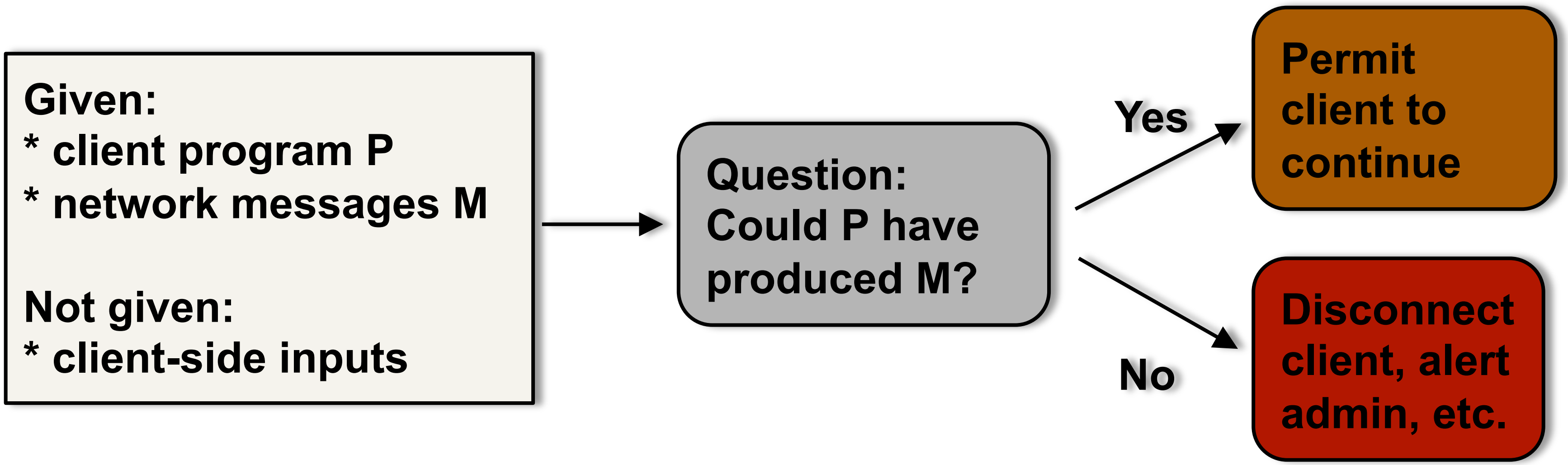


How Can We Defend Tenant Servers?



- Client validation: permit authorized client software only
  - Eliminates entire classes of attacks *without knowing about them*
  - Usually requires client modification or sending of client inputs
- Run for inline defense, or offline for rapid detection of exploit attempts

Client Behavior Validation
[Chi, Cochran, Nesfield, Reiter, Sturton; 2016]

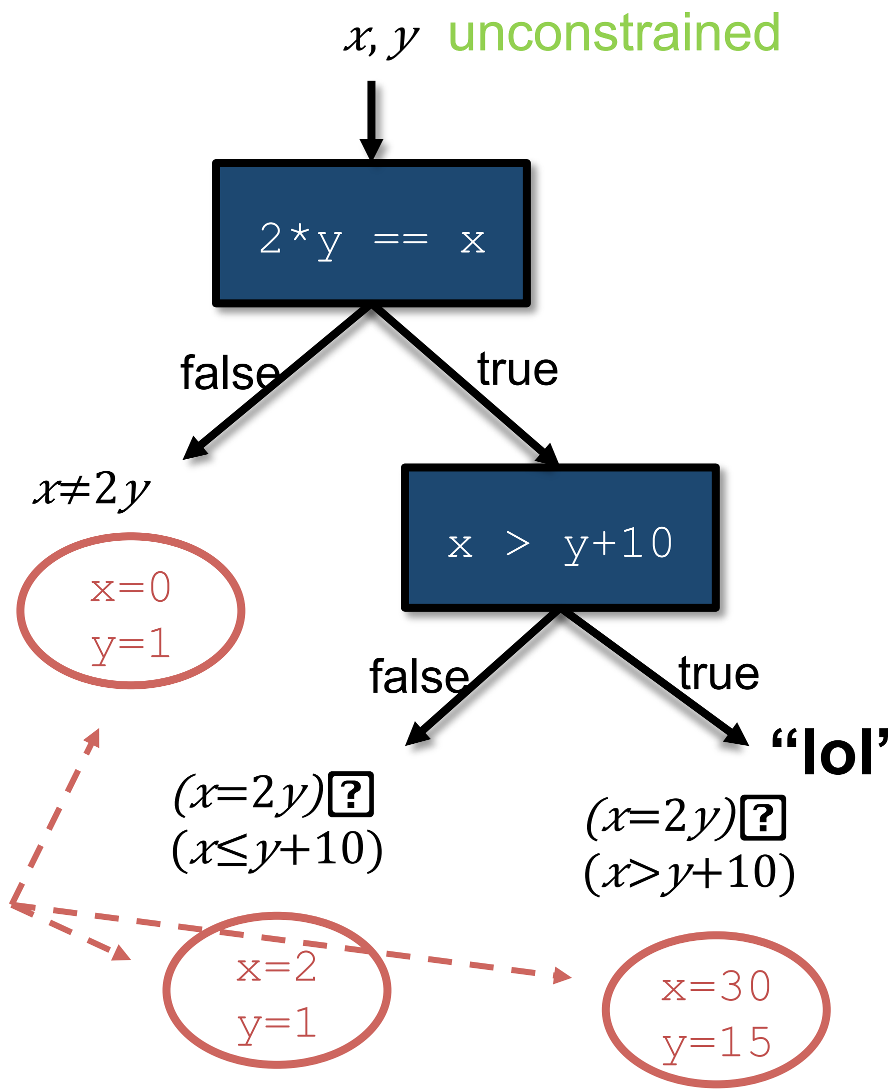


- General case: undecidable
- Specific instances may be practical

Symbolic Execution

```
x = sym_input();
y = sym_input();
testme(x,y);

void testme(int x, int y)
{
    int z = 2*y;
    if (z == x) {
       if (x > y+10)
           printf("lol");
    }
}
```



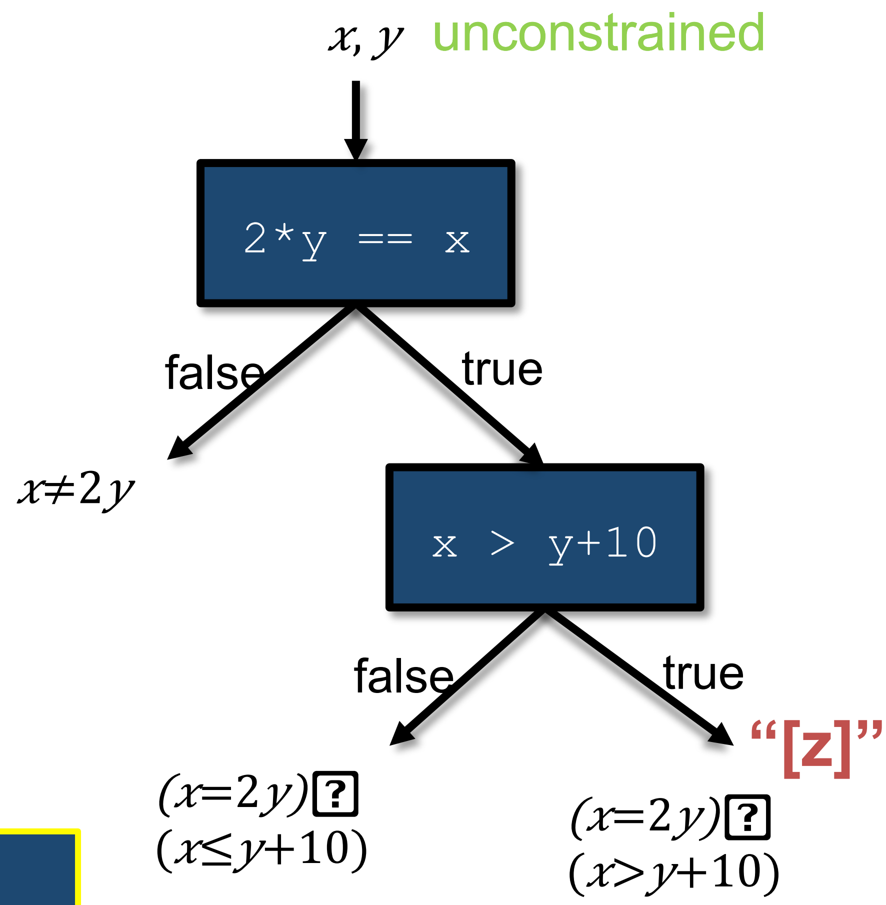
Apply SAT solver to obtain concrete test cases.

Example adapted from: Cristian Cadar, and Koushik Sen. "Symbolic execution for software testing: three decades later." Communications of the ACM 56.2 (2013): 82-90.

Symbolic Execution

```
x = sym_input();
y = sym_input();
testme(x,y);

void testme(int x, int y)
{
     int z = 2*y;
     if (z == x) {
        if (x > y+10)
             send(z);
     }
}
```

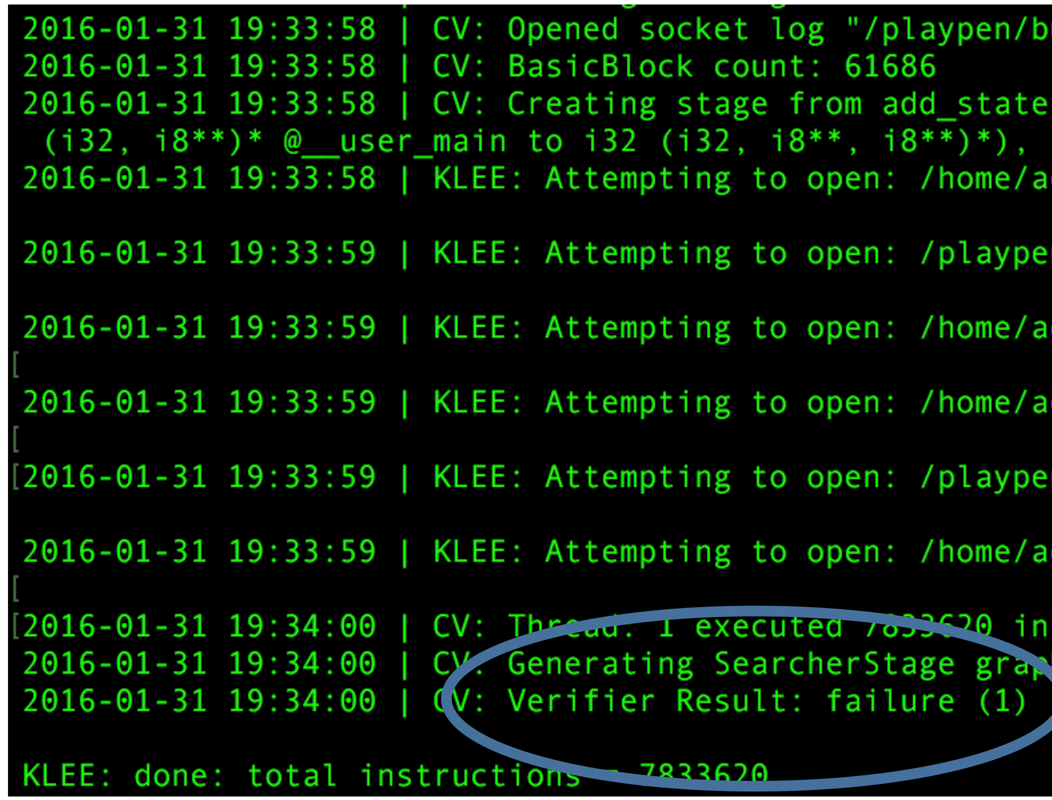


Can this program produce…

- **42?** **Yes** ($x=42$, $y=21$)
- **41?** **No** ($z=2y$ so it must be even)
- **2?** **No** ($x>y+10$ is violated)

Example: Detecting Heartbleed (Without Looking For It)

- Malicious s_client
  - performs handshake
  - sends Heartbleed exploit
- Validation
  - Handshake is verified
  - No explanation found for malicious Heartbeat



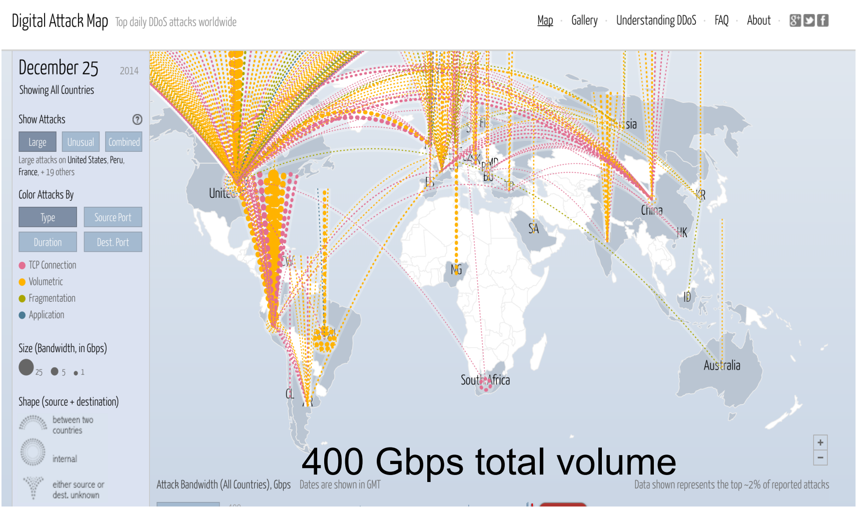
Detection in ~2s

Performance

- Verification latency is not (yet) fast enough for inline verification in latency-sensitive apps
- It can, however, keep pace with many common applications
  - Example: In our experience, OpenSSL and BoringSSL behavior in Gmail connections can be verified during the connection

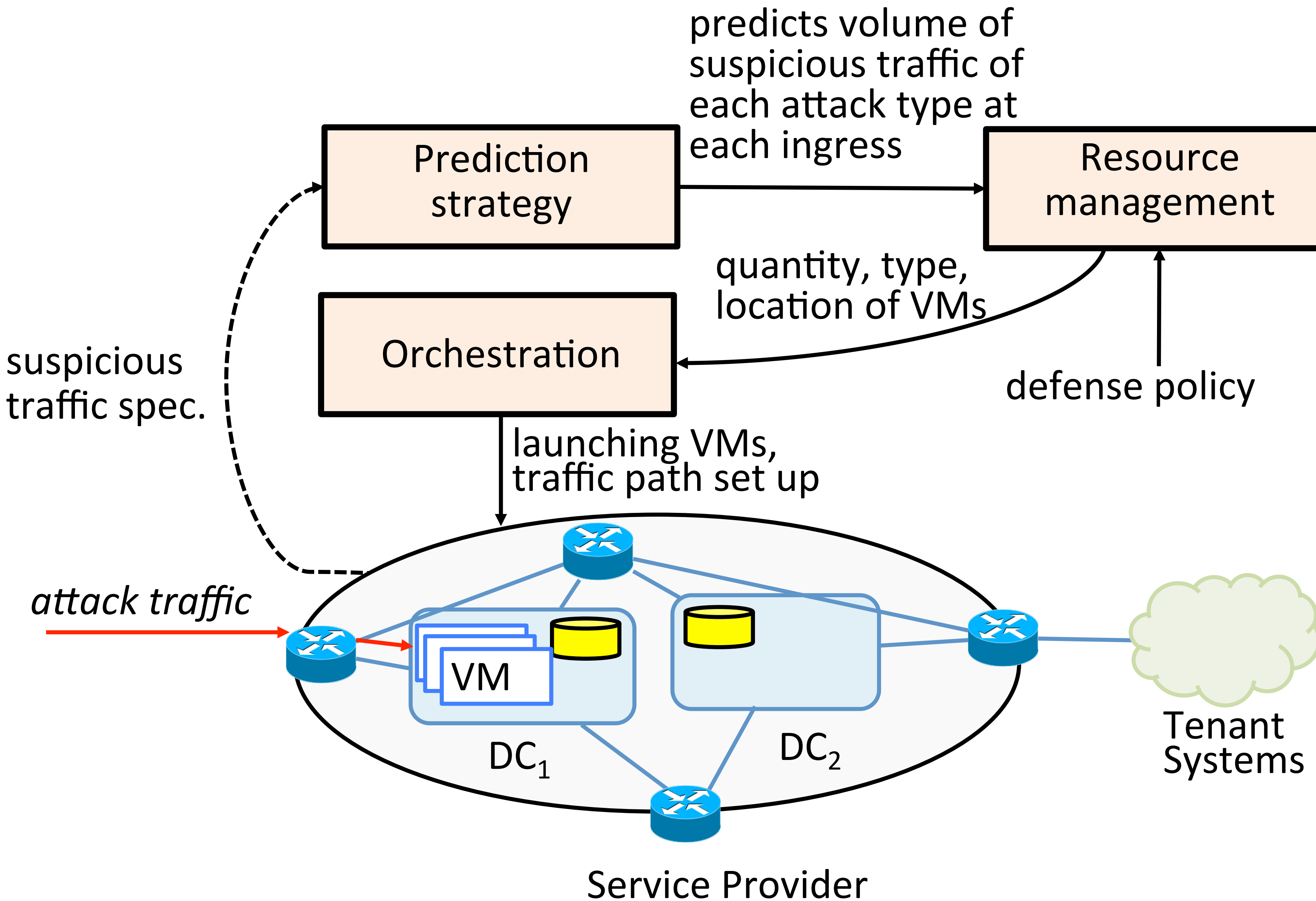
DDoS Defense: Bohatei
[Fayaz, Tobioka, Sekar, Bailey; USENIX Sec. 2015]

- DDoS attacks a persistent problem
- Today's defenses involve proprietary hardware
  - Expensive
  - Fixed: capacity, functionality, location

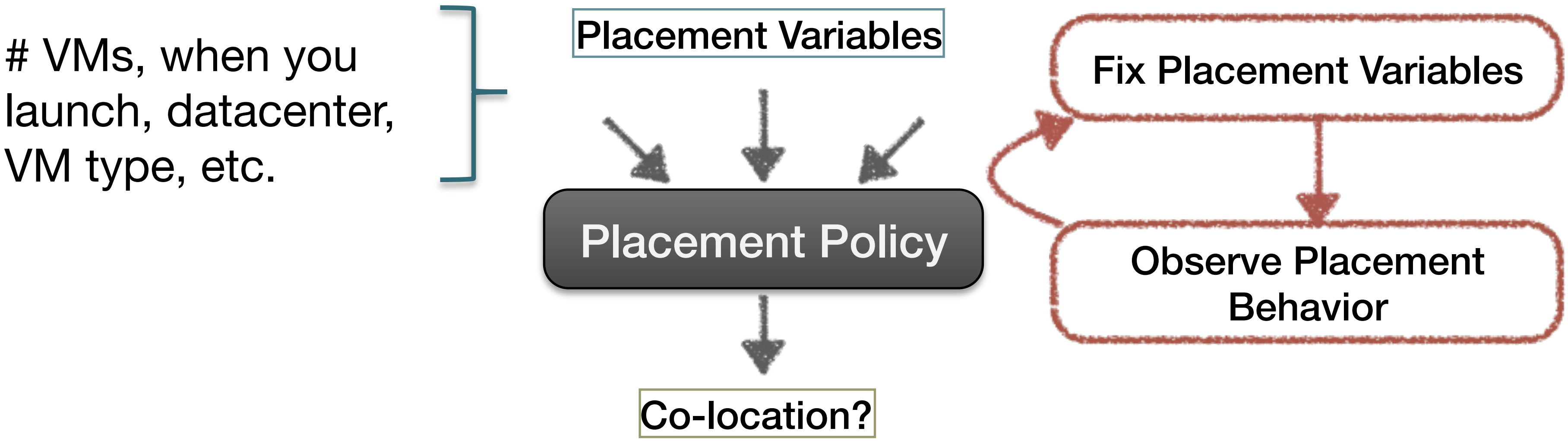


- Bohatei is a cost-effective, low-latency, agile DDoS defense by provider for tenants
  - manages dynamic 500 Gbps DDoS against tenant with < 1 min. reaction time

DDoS Defense: Bohatei
[Fayaz, Tobioka, Sekar, Bailey; USENIX Sec. 2015]



Side Channels: A Co-Location Vulnerability Study
[Varadharajan, Zhang, Ristenpart, Swift; USENIX Security 2015]



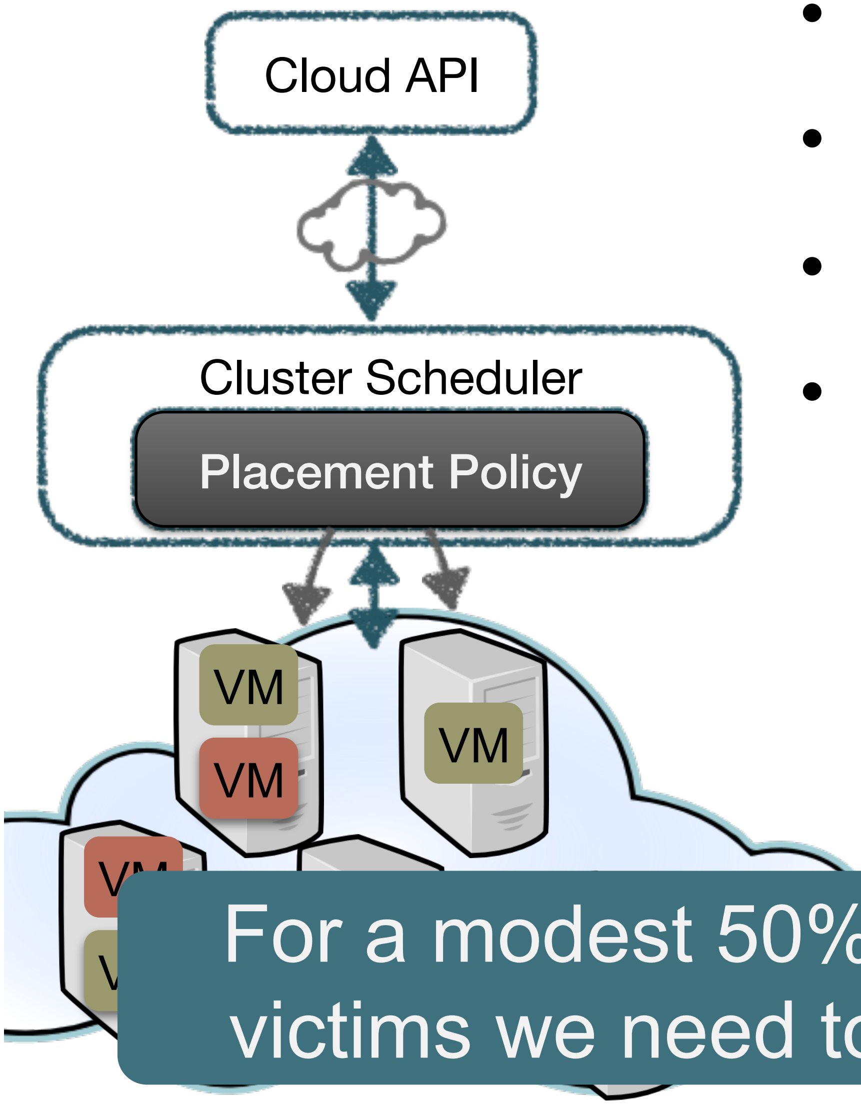
Study spanning 3 months & exploring 6 placement variables

Google Compute Engine amazon web services™ EC2 Microsoft Azure

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?

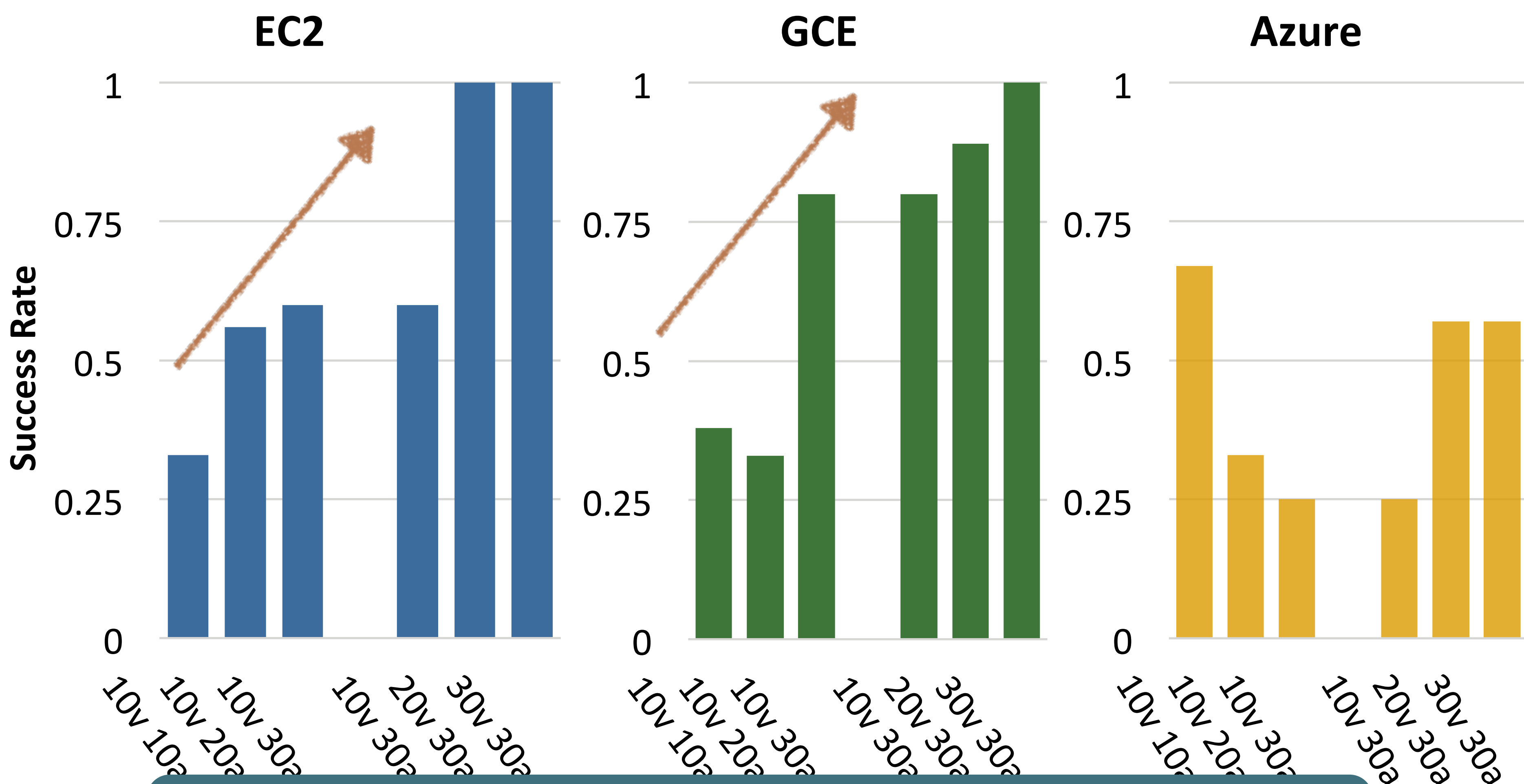


- Random placement policy
- N = 50,000 machines [re:Invent'14]
- *v* - victims and *a* - attacker VMs
- Probability of Collision:
  $P_c = 1 - (1 - v/N)^a$

v	a = ln(1− P_c)/ln(1−v/N); P_c= 0.5
10	3466

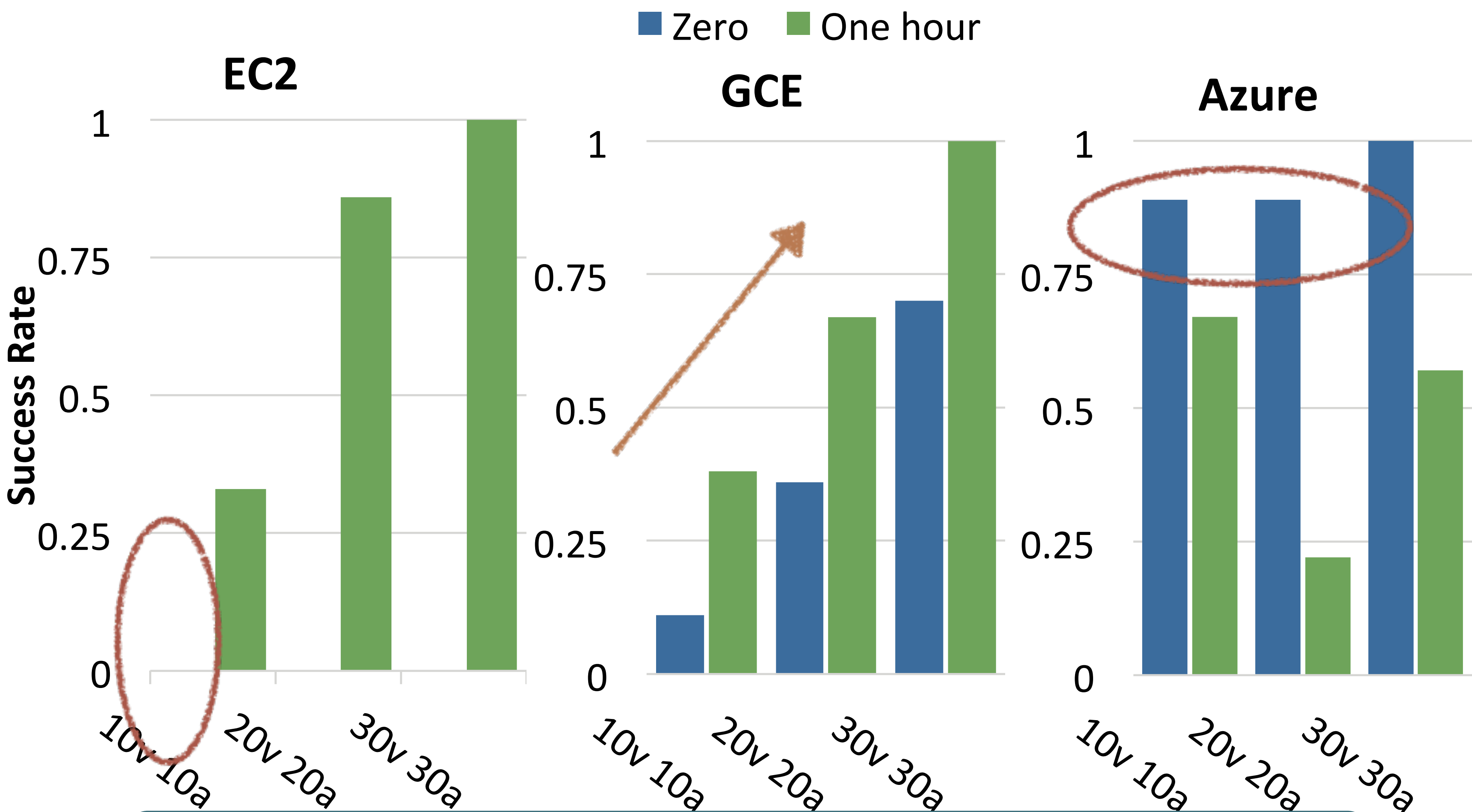
For a modest 50% success rate with 10-30 victims we need to launch 1000-3000 VMs

Results: Varying Number of VMs



Co-location is possible with as low as 10 VMs and always achieve co-location with 30 VMs

Results: Varying Delay between Launches



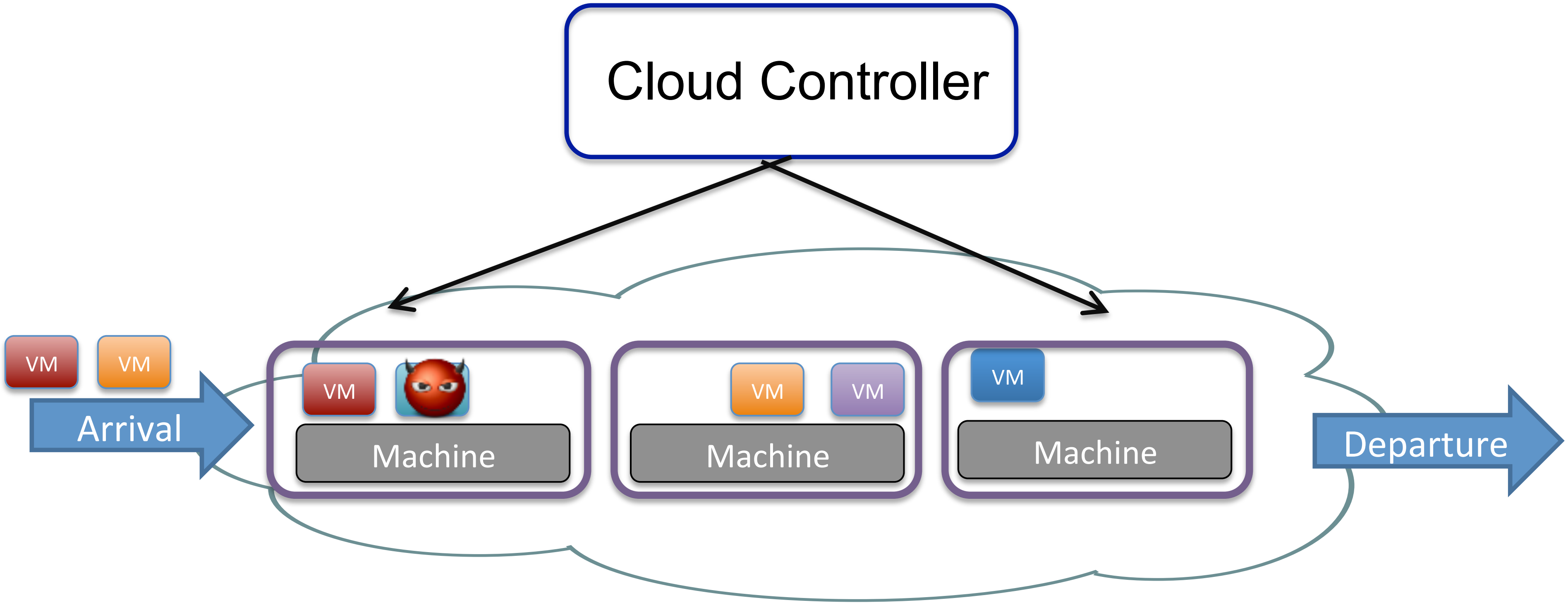
Placement policy for each cloud significantly varies

Side-Channel Defense

- A primary concern with co-location vulnerabilities is side channels
- Goal: a defense against side channels that is
  - *General* across a broad spectrum of side-channel attacks
  - *Immediately deployable* with minimal or no modifications to existing cloud hardware and software

Key idea: Migration

Tackle the *root cause* of side channels



Leverages the cloud provider as a trusted ally via an *opt-in* migration-as-a-service

Side-Channel Defense: Nomad

[Moon, Sekar, Reiter; CCS 2015]

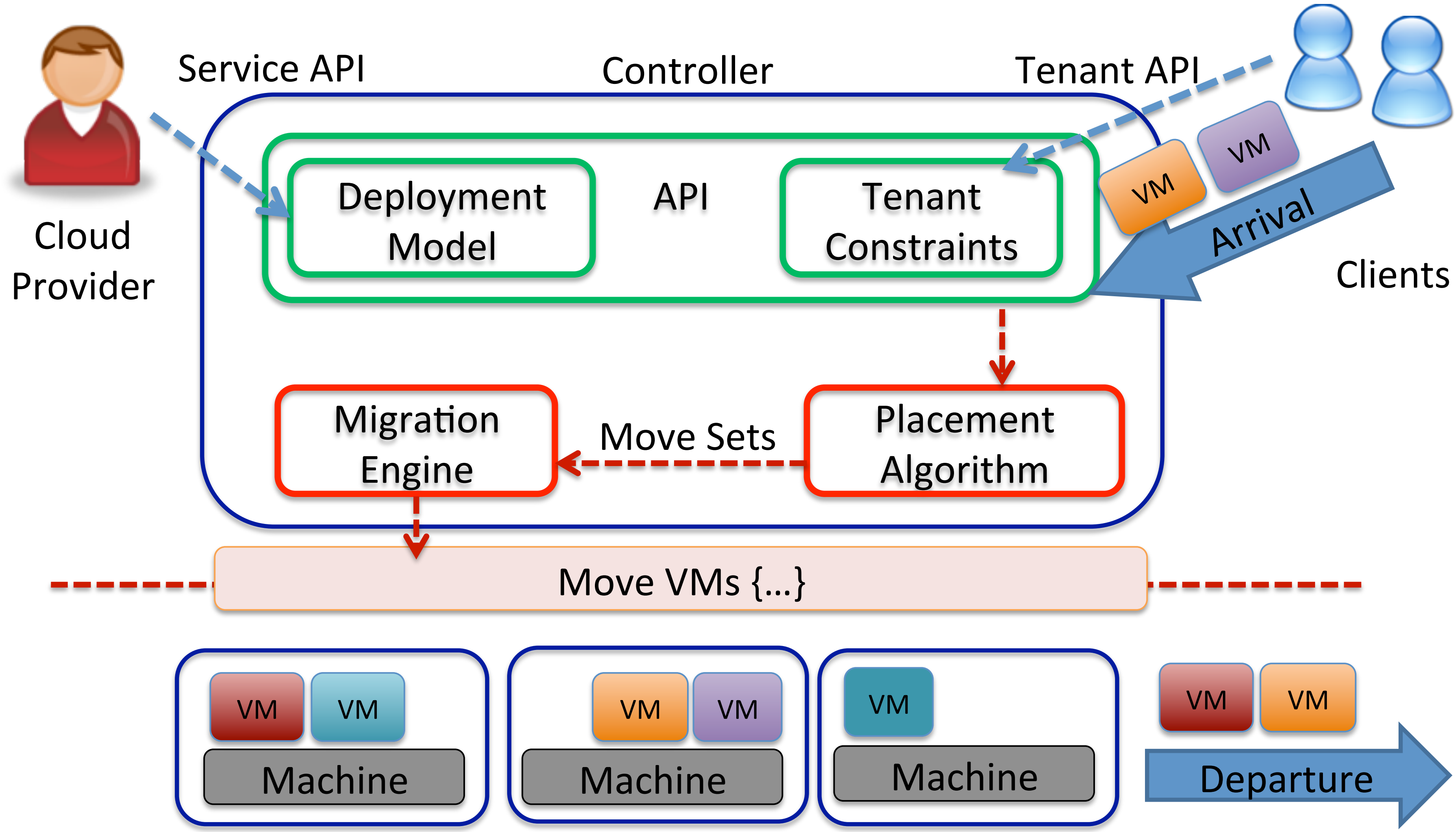
1) Vector-Agnostic Defense

Agnostic to the specific side-channel vector used

2) Minimal Modification

Can be deployed "out of the box"; requires only changing the VM placement algorithms

Nomad Overview

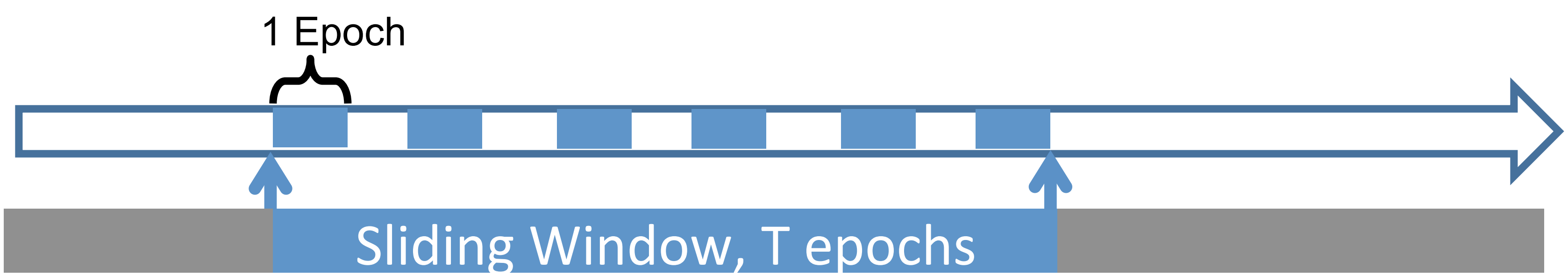


Threat Model

- Adversary capabilities
  - Identity unknown
  - Arbitrary side channels
  - Can identify targets
  - Arbitrary workloads
  - Efficient information collation
- Adversary limitations
  - No control over VM placement
  - No collusion among clients (i.e., Sybil attack)

Information Leakage Model

- What is the effect of co-residency on the amount of information leakage?
- Three dimensions
  1. Over time

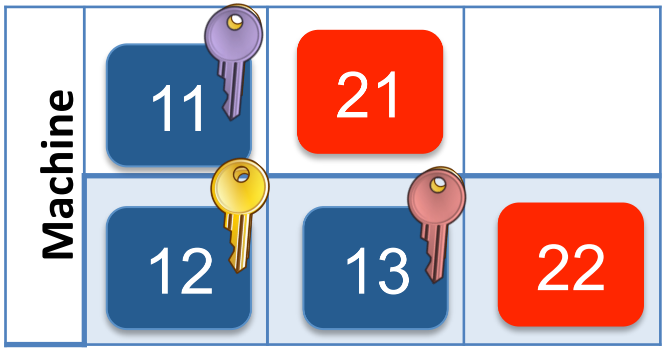


Extent of *information leakage* $\propto$ Number of epochs that VMs are co-resident in a sliding window of T epochs

Information Leakage Model

- What is the effect of co-residency on the amount of information leakage?
- Three dimensions
  2. Over victim VMs

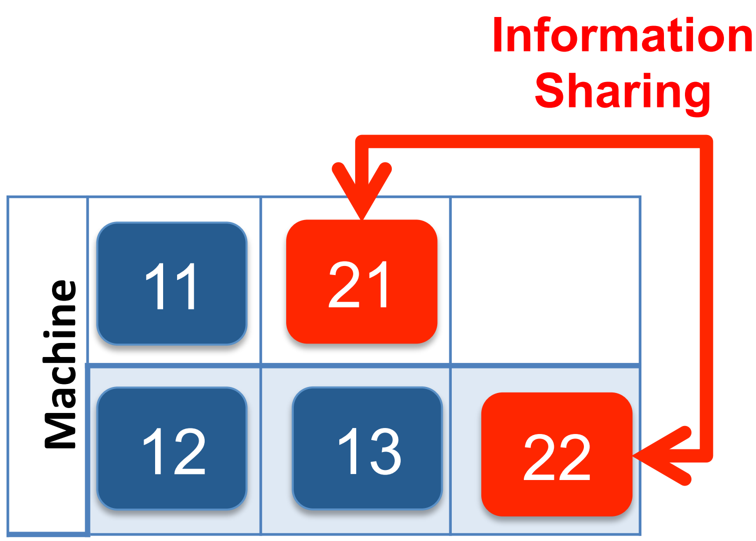
Replicated
vs.
Non-replicated



Information Leakage Model

- What is the effect of co-residency on the amount of information leakage?
- Three dimensions
  3. Over adversary VMs

Non-collaborating
vs.
Collaborating

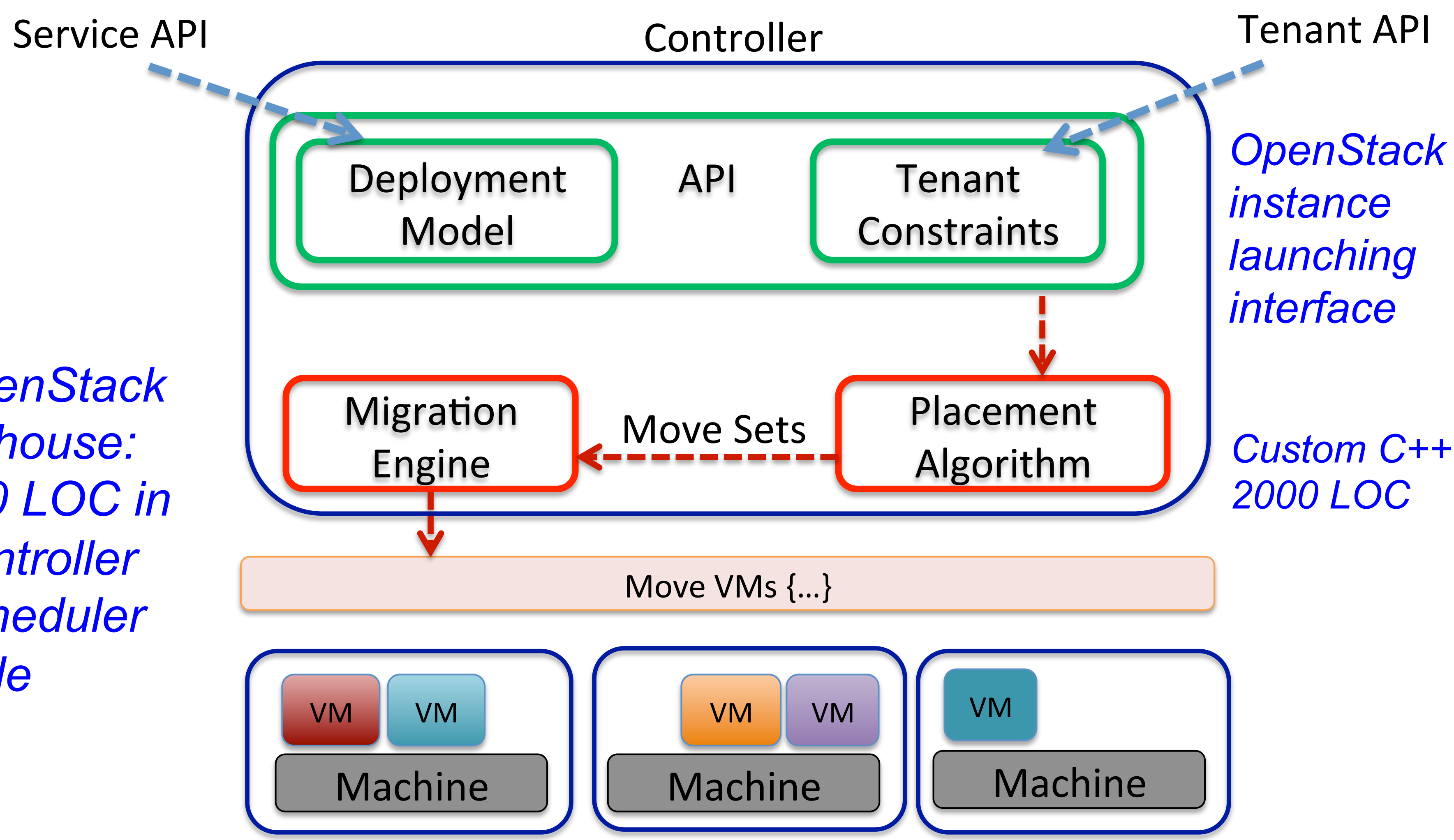


Nomad Placement Algorithm

- Nomad migrates VMs so as to (approximately) minimize information leakage over a sliding window
  - Subject to a fixed migration budget
  - Perfectly minimizing leakage isn't tractable (ILP)
- Nomad placement algorithm is greedy, but even then, requires a number of optimizations to be scalable
  - Limit migrations to free-inserts or 2-way swaps
  - Hierarchical placement: partition machines into clusters, and map tenants to clusters
  - Use lazy and incremental evaluation where possible

Nomad System Implementation

OpenStack Icehouse: 200 LOC in Controller Scheduler code



Nomad Evaluation Summary

- Greedy algorithm limits information leakage nearly optimally (albeit heuristically)
- Nomad is scalable
  - Cluster size can be 1,500 to handle 1 min goal
  - For cluster size of 20
    - Nomad takes 0.015s
    - ILP takes > 1 day
- Migrations do not substantially hurt job performance

The Driving Vision

- A "cloud control platform" that supports
  - Improved cloud and tenant security
  - Innovative services to enable new modes of tenant interaction

- … through new tech for better managing
  - Tenants' clients (credentials, protocols, …)
  - Tenant infrastructure (outsourced services, …)
  - Tenant-to-tenant ecosystem (trust management)

For more information, please see http://silver.web.unc.edu

Questions?