8271 discussion of cloud computing security (combined)

Stephen McCamant

University of Minnesota

Outline

Get Off of My Cloud

Administrative discussions

Multi-Cloud Oblivious Storage

Old and new topics in security

- Paper type 1: new idea, never been done before
  - Main contribution is novelty
  - Incentive to be first, maybe even a race
- Paper type 2: improvement in an already-busy area
  - Contributions judged differentially
  - Incentive to optimize

Cloud threats, old and new

- Old: your system's regular vulnerabilities
- New but understood: need to trust cloud provider
- Focus here: attacks from cloud neighbors

Case study: Amazon EC2

- Largest, highest-profile infrastructure cloud provider
- World-spanning data centers, instance sizes $0.02-$6.82 per hour
- Many instance types use Xen to multiplex one physical machine

Ethical/legal sidebar

- Important for academic researchers to do things "by the book"
- Ethical obligations may be greater or less than legal ones
- Here: CFAA, EC2 user agreement

Placement and extraction

- *Placement*: get an instance on the same physical machine as the victim
- *Extraction*: given placement, get confidential info

Network probing

- TCP traceroutes, port 80 and 443 scans, DNS resolution
- Instances have one name, but separate public and internal IP addresses

Network mapping

- Internal addresses reflect topology
- Disjoint by availability region, clustered by instance type
- Dom0s in an adjacent block

Network-based co-residence checks

- Dom0 in traceroute (easiest)
- Close IP addresses
- Smallest packet round-trip times
- All found to have "effectively zero" false positives

Hard disk usage channel

- Measure contention for hard disk (e.g., seek times) between VMs
- "No attempt to optimize" bandwidth: 0.0005 bits/sec (33 mins per bit)
- Why so slow?

Covert channels and side channels

- "Covert channel": generally send and receiver cooperate
  - One classification: storage channels, timing channels
- "Side channel": "sender" is passive victim
  - Can again include timing, also error messages, power usage, etc.

Observed placement locality

- Sequential locality: new instance likely to use same machine as old dead one
- Parallel locality: instances started close in time more likely to share
- Non-locality: one account never given two instances on same machine

Evaluating brute-force placement

- Chose 1686 victims
  - Small instances in zone 3 with public web servers
- Launched probe instances and checked co-residence
  - 510 probes: hit 127 victims
  - 1785 probes: hit 141 victims, 8.4%

Using locality

- Idea: use parallel locality, try to start probes soon after victim
  - Perhaps can trigger victim start, such as if it's based on demand
- About 40% coverage for 20 victims and 20 probes
- Also demonstrated against demos of commercial services

Cache: Prime+Trigger+Probe

1. (Prime) Fill cache with my data
2. Busy loop until preempted (recognize with TSC)
3. Measure time to re-read my data

- Must play tricks to defeat CPU pre-fetch
- Differential coding to resist noise

Load and traffic estimation

- Check for co-residence using system load as a covert channel
- Estimate traffic load on co-resident web server

Keystroke timing attack (classic)

- Fine-grained keystroke timing can reveal information about text typed
- Especially given per-user training
- Demonstrated in lab against passwords typed over SSH, without breaking crypto
  - 50× speedup over exhaustive search

Keystrokes in Xen

- Lab installation with CPU pinning, otherwise idle; not real EC2
- Threshold cache activity level
  - More than idle, less than otherwise busy
- 5% false negatives, 0.3 false positives per second
- Timing resolution 13ms, enough for prior attacks

Countermeasures: limited

- Randomize and isolate network structure
  - Timing measurements still possible
- Block or add noise to covert channels
  - Hard, and how to know you have them all?
- Avoid locality in placement algorithm
  - Reduces but does not eliminate attacks

Countermeasure: pay for isolation

- Pay extra to have machines all to yourself
- Argument: fair cost upper-bounded by cost of one physical machine
- Not implemented
  - Though compare: GovCloud

Outline

Get Off of My Cloud

Administrative discussions

Multi-Cloud Oblivious Storage

Next week: Bitcoin

- For Monday: double-spending attacks
- For Wednesday: real anonymity with Zerocoin

Choosing presentation topics

- I still need to post more papers
- Is volunteering viable?
- Possible alternative: lottery plus trading

Choosing project topics

- Start looking for groups and topics now
- Meet with me next week or week after
- Proposals due February 28th (less than one month)

Outline

Get Off of My Cloud

Administrative discussions

Multi-Cloud Oblivious Storage

Motivation: hide access patterns

- Information is leaked by what you access when
- Consider encrypted email, medical info, etc.
- Goal here: conceal location, read vs. write

What's revealed by plain encryption?

- Imagine we encrypt every disk block with function E
- Adversary can still see patterns of locations
- If $b_1 = b_2$, $E(b_1) = E(b_2)$

Using probabilistic encryption

- Probabilistic encryption: randomized, returns different ciphertext each time
  - Standard in public key, theory, and with modes of operation
- To conceal read vs. write, always replace block with new encryption

Straw man 1: access every block

- For each virtual access (read or write), access (read and write) every physical block
- Secure, but impractical

Straw man 2: shuffle all blocks

- Use pseudo-random permutation to shuffle all block locations
- Secure if you never access a block more than once
  - But leaks on any repeated operations
  - Can't have, e.g., read after write

Goldreich square-root construction

- First semi-practical idea (STOC 1987)
- Cache of $\sqrt{m}$ locations accessed each time, plus shuffled copy
- Dummy accesses for consistency
- Reshuffle after $\sqrt{m}$ operations

G&O hierarchical idea

- Split into levels of exponentially increasing size
- Write back in smallest level, then reshuffle into larger
- Various kinds of hashing can be used
- Polylog amortized cost for $O(1)$ client storage
  - But still pretty impractical

The client bandwidth constraint

- In many storage outsourcing applications, major constraint is client's network bandwidth
- Client has significant local storage
  - Not enough for all data
  - But enough for an index (order of one word per block)

Multi-cloud approach

- Cloud-to-cloud bandwidth more than client-to-cloud
- Use multiple (e.g. 2) clouds
- Require: not all clouds are malicious
- Major savings, especially on client bandwidth

Threat models in protocols

- (Fully) honest: follows the protocol exactly
- Malicious: can do anything (worst case)
- Semi-honest, AKA honest-but-curious: follows protocol, but may try to learn secrets from seen data

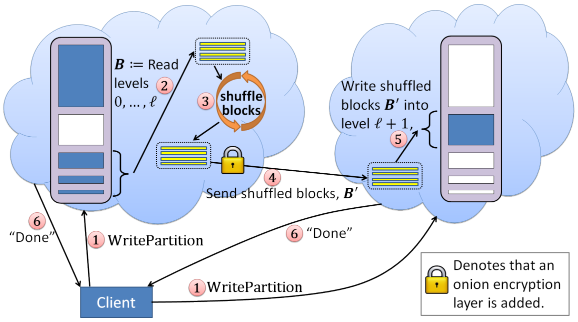
SSS partitioning

- Divide data into $\sqrt{m}$ partitions of size $\sqrt{m}$
- Client keeps location index and $\sqrt{m}$ blocks of cache
- Improves worst-case and constant factors, but still needs log $\sqrt{m}$ (e.g., 10×) accesses to read

Splitting between clouds

- Make expensive operations cloud-to-cloud
- Do operation in one cloud to hide from the other
  - "Non-colluding" confidentiality assumption

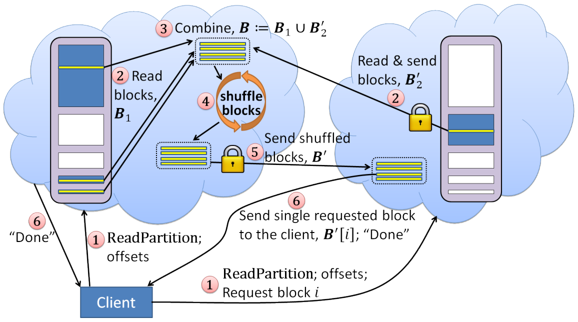
Write: oblivious shuffling



B := Read levels 0, …, ℓ
2
3
shuffle blocks
4
Send shuffled blocks, B'
Write shuffled blocks B' into level $\ell + 1$,
5
6
"Done"
1
WritePartition
6
"Done"
1
WritePartition
Client
Denotes that an onion encryption layer is added.

Figure source: taken from the paper

Read: oblivious selection



3
Combine, $B := B_1 \cup B'_2$
2
Read blocks, B_1
4
shuffle blocks
Read & send blocks, B'_2
2
5
Send shuffled blocks, B'
6
Send single requested block to the client, $B'[i]$; "Done"
6
"Done"
1
ReadPartition; offsets
1
ReadPartition; offsets; Request block i
Client

Figure source: taken from the paper

Homomorphic checksum

- Linear checksum allows computation on encrypted blocks
  - Note: not secure after an adversary has seen examples!
- Combined with PRF (imagine: MAC) and authenticated encryption

Experimental deployment

- Amazon EC2 (AWS) with SSDs
- Microsoft Azure, lacking SSDs
- Up to 5 servers (max out client bandwidth)
- $3.10 per hour plus $2.50 per GB for one server

Bottleneck analysis

- Client bandwidth 2.6×
  - Compare 2× for read and write
- In practice: Azure's non-SSD disk speeds
- Assuming SSDs, double throughput up to 6MB/s
  - Based on cloud-to-cloud bandwidth bottleneck, 30-60MB/s