Data Confidentiality in the Cloud: Laser Gunfight at the O.K. Corral?
Approaches to stopping the malicious insider at the cloud provider

Miguel Correia
IST / INESC-ID

Session ID: CLD-108
Session Classification: Intermediate

Cloud computing in a nutshell

- Computing as a utility
- Pay-as-you-go / pay-per-use
- Resource pooling
- Elasticity
- Large-scale datacenters
Talk is about IaaS and public clouds

- Infrastructure as a Service (IaaS): the service provided are virtual machines, storage
  - e.g., Amazon EC2, Amazon S3
- Public cloud: the cloud provider and cloud user are different companies

Security in the cloud (from the user viewpoint)

- Challenges
  - The system is no longer in the user premises
  - The infrastructure is shared with other users
  - The access is made through the internet
- The three classical security attributes can be jeopardized: confidentiality, integrity, availability
Outline

- How to steal data in the cloud
- Approach 1: improve the infrastructure
- Approach 2: build a cloud-of-clouds
Malicious insider and confidentiality

- The data is in the cloud and the malicious insider is a real problem
  - CyberLynk (March’09) and Google (early’10) events

Basic cloud architecture (IaaS)

- Service provided is the execution of Virtual Machines (VMs): full sw stack, including OS
- Servers run an Hypervisor (or VMM) that supports the execution of several VMs
Experiments

- We played the role of a malicious insider with access to the management VM
- The “cloud” was just a single machine
  - Hypervisor was Xen
  - Management VM was Xen Dom 0 with Linux (Ubuntu)
  - 1 user VM (victim) with Linux and an Apache server

Attack 1: steal passwords in memory

- Trivial: take mem snapshot, look for passwords
  
  `$ xm dump-core 2 -L lucidomu.dump`
  
  `Dumping core of domain: 2 ...`
  
  `$ cat lucidomu.dump | strings | grep loginpwd`
  
  `loginpwd`
  
  `$ cat lucidomu.dump | strings | grep apachersapwd`
  
  `apachersapwd`
Attack 2: steal private keys in memory

- Trivial: they’re in a standard format in memory
  
  ```bash
  $ xm dump-core 2 -L lucidomu.dump
  Dumping core of domain: 2 ...
  $ rsakeyfind lucidomu.dump
  found private key at 1b061de8
  version = 00
  modulus = 00 d0 66 f8 9d e2 be 4a 2b 6d be 9f de
  46 db 5a
  ...
  publicExponent = 01 00 01
  privateExponent = ...
  prime1 = ...
  prime2 = ...
  ```

Attack 3: steal files in file system

- Trivial: essentially mounting a drive (with LVM)
  
  ```bash
  $ lvcreate -L 2G -s -n lv_st /dev/main_vol/domu
  Logical volume ‘lv_st’ created
  $ kpartx -av /dev/main_vol/lv_st
  ...
  $ vgscan
  Search for LVM volumes
  Found volume group ‘LucidDomU’
  $ vgchange -ay LucidDomU
  Activate the snapshot volume
  $ mount /dev/LucidDomU/root /mnt/
  ```
Current solutions?

- “Cloud Computing Roundtable” (Nov/Dec 2010)
  - senior staff from: Google, Microsoft, Cisco, Amazon, Cloud Security Alliance
- “We have very strict procedures in place for when our employees are allowed to [physically] access the machines the customer data resides on.”
  - But the attacks we saw can be done remotely
- “We keep track of every action that they take on those machines, and we log all that information for later audits”
  - But detecting later can be too late
- “We have zero tolerance for insiders abusing that trust”

Cryptography?

- Obvious solution: simply encrypt the data
- But what is data in IaaS?
  - User files, web pages, databases, variables, data structures, etc.
  - Is it possible to modify applications to handle encrypted data? An application server (Tomcat, JBoss, …)?
  - Where do we store the encryption keys safely?
- Moreover applications often manipulate data
  - Manipulate encrypted data: fully homomorphic encryption
  - Slow and does not work with data from several clients
Approach 1: improve the infrastructure

Key idea

- To prove to the cloud user that its data is in a server with a “good” software configuration
  - e.g., in which the management VM has no snapshot function
- Do it with the Trusted Platform Module (TPM)
  - a security chip designed by the Trusted Computing Group, now shipping with common PC hardware
TPM basic functions

- Two basic functions:
  - Storage of cryptographic keys – e.g. to protect RSA private keys from disclosure
  - System software integrity measurement – to do certain operations (or not) depending on the software running

Measurements and PCRs

- TPM has at least 16 Platform Configuration Registers (PCR)
- A PCR stores (typically) a measurement of a software block, i.e., its cryptographic hash
  - During system boot, the i\(^{th}\) module to run stores the hash of the (i+1)\(^{th}\) module in PCR\(_{i-1}\)
  - Example: BIOS stores \texttt{hash(boot loader)} in PCR\(_{0}\); boot loader stores \texttt{hash(hypervisor)} in PCR\(_{1}\)
- A vector of PCR values gives a trusted measurement of the software configuration
Measurements and PCRs (cont)

- Can't the 1st module provide a false hash of the 2nd?
- We assume we can trust the 1st module, thus called the Static Root of Trust for Measurement (SRTM)
- Can't a PCR be overwritten at any time?
- No, there is no write operation, only extend
  - \( \text{PCR}_i \leftarrow H(\text{PCR}_i \ || \ h) \) (the 1st time, \( \text{PCR}_i = 0 \))
  - After the 1st extend, it's infeasible to store exactly \( 0||h \) in \( \text{PCR}_i \) (due to properties of cryptographic hash functions)

Remote attestation

- Computer gives to challenger a measurement of the software configuration, i.e., a vector of PCR values
  - Challenger has the Endorsement Key Certificate, signed by the TPM vendor (means it's a real TPM!)
  1. Request attestation
  2. Request TPM vector of PCRs signed with EK
  3. PCR vector (signed with EK)
  4. Verify signature and if PCR values match a trusted configuration
Solution overview

- Servers run a Trusted Virtualization Environment (TVE), formed by hypervisor + management VM that the user trusts
- TVE does not provide dangerous operations to administrators: snapshot, volume mount
- TVE provides only trusted versions of certain operations: launch, migrate, backup, terminate VMs
- VMs enter and leave a TVE encrypted
- Users do remote attestation of TVEs/operations to be sure that their VMs are either in a TVE or encrypted

Trusted virtualization environment

- The virtualization environment is measured
  - At boot time, hashes of the software components that are loaded are stored in PCRs
  - At least: boot record, hypervisor, management VM (kernel, management software)
- The environment is a TVE if its measurements (PCR values) fall in a set of TVE-configurations
Open problems

- Gap between checking a measurement (just a hash) and trusting a complex software module
  - How can we know that there aren’t vulnerabilities, undesirable functionality or malware inside?
- Putting this solution in production is far from simple
  - Short time to market and too many players: cloud provider, software producers, assurance labs

Approach 2: build a cloud-of-clouds
Securing the cloud

- 1st solution: improve the cloud infrastructure with trusted computing
- 2nd solution: build a (virtual) cloud-of-clouds based on a few clouds – DepSky system
- First can be implemented by providers, second by users

Cloud-of-clouds’ benefits

- Can tolerate data corruption
  - Due to malicious insiders, other attacks, accidental faults (e.g., due to bugs)
- Can tolerate datacenter and cloud outages
- No vendor lock-in
- Faster read access
- Confidentiality…
Cloud-of-clouds object storage

- No longer IaaS cloud computing, (only) storage
- Cloud-of-clouds provides the same service as single cloud: read data, write data, etc.

Write protocol

Assume a version of the file is already stored
Read protocol

Solution so far

Provides availability despite data corruption and cloud outages but:
1. Data is accessible by cloud providers
2. Requires \( n \times |\text{Data}| \) storage space
Combining erasure codes and secret sharing

Only for data, not metadata

Data encrypt

Data

F1 F2 F3 F4

S1 S2 S3 S4

Cloud A Cloud B Cloud C Cloud D

Key

Key

Encrypted so data can’t be read at a cloud!
Only twice the size of storage, not 4 times!

Performance evaluation setup

- Prototype: 3K LOCs (Java), REST/HTTPS
- Experimental setup
  - 2 DepSky versions: A (availability), CA (availability+ confidentiality)
  - 4 commercial storage clouds: S3 (Amazon S3), WA (Windows Azure), NX (Nirvanix SDN) and RS (Rackspace)
  - Clients in 8 sites around the world (PlanetLab)
  - 437K+ reads/writes in Sep./Oct. 2010
DepSky storage costs ($)

DepSky-CA storage cost (1M DU) ≈ 2 × (average cloud cost)

DepSky latency (100KB DU)

DepSky read latency is close to the cloud with the best latency

DepSky write latency is close to the cloud with the worst latency
Conclusions (1)

- Cloud security undeniable problem for organizations that want to use it for critical systems/data
- The malicious insider is an especially hard problem
- Two approaches, but not exactly for the same problem
Conclusions (2)

- Approach 1 – improve the cloud infrastructure with trusted computing
  - Cloud providers may implement something of the kind
  - But too many open problems yet
- Approach 2 – build a storage cloud-of-clouds based on a few clouds – DepSky system
  - A user-side solution, so easier to deploy
  - More expensive than single cloud, but not excessively

Apply Slide

- In the next three months you should:
- Identify critical data your company has in the cloud
- If your company uses the cloud for computing
  - Identify hypervisor/management VM used
  - Ask provider operations supported by the mgmt VM
  - Ask provider what protections from admins are used
- If you company uses storage clouds
  - Consider encrypting data and using two clouds
- In one year: follow cloud evolution; use DepSky?
Thank you!
More info:
google miguel correia inesc-id