Software Security in the Clouds

Miguel Pupo Correia
Universidade de Lisboa, Faculdade de Ciências, LASIGE

2nd Ibero-American Web Applications Security Conference
IBWAS’10

Outline of the talk

• Motivation
• Security challenges in the cloud
• Software security in the cloud
• Further security: distributing trust
• Conclusions
Cloud computing

- Fundamental ideas
  - Computing as a utility
  - Pay-as-you-go (public cloud)
  - Resource pooling
  - Elasticity

- Implementation
  - Large-scale datacenters
  - Cloud provider vs cloud users

Cloud computing

Service models:

- **Infrastructure as a Service (IaaS):** virtual machines, storage (e.g., Amazon EC2, Windows Azure)

- **Platform as a Service (PaaS):** programming and execution (e.g., Google AppEngine, Force.com)

- **Software as a Service (SaaS):** mostly web applications (e.g., Yahoo! Mail, Google Docs)

- Web is crucial in PaaS and SaaS – role of OWASP?
Security in the cloud?

• Recall the three attributes
  – Confidentiality – no disclosure of data to unauthorized entities
  – Integrity – no unauthorized modifications of the system or data
  – Availability – readiness of the system to provide its service

• The three are important in the cloud

• Challenges
  – The system is no longer in the organization premises
  – The system is shared with other users
  – The access is through the internet

SECURITY CHALLENGES IN THE CLOUD
Unavailability

- Problems in the Internet – relatively frequent
  - Congestion
  - Problems in the client or ISP equipment (routers, etc.)
  - More global problems (Cisco bug + RIPE NCC test Aug. 2010)
- Problems at the cloud (e.g., Google AppEngine)
- Denial of service attacks (e.g., Amazon 2009)

Loss and corruption of data

Can happen in the cloud as anywhere else

- Danger Inc. / Sidekick lost contacts, notes, photos etc. of its clients; took days to recover them (Oct. 2009)
- Ma.gnolia lost all data from all clients, half TB (Feb. 2009)
Privacy/confidentiality violation

- Data is in the cloud provider machines
  - The provider may be trusted; there are legal defenses; but
- There can be a malicious insider
  - Can capture passwords, private keys, software, etc.
  - Not specific in the cloud, but the cloud operators are unknown/…
- Demo of operator/sysadmin capturing private keys
  - Basic cloud environment emulation: Xen hypervisor
  - Dom-0, Dom-1
  - Video
  - Only 2 commands needed!

Attacks via management interface

- In the cloud the attack surface is expanded with the cloud management interface
  - Control/monitoring of virtual machines, users, etc.
  - Web console, web services, REST
- Attacks through the interface
  - Vulnerabilities that allow personification of legitimate user: SQLI, XMLI, XSS, CSRF, etc.
  - Microsoft, “Secure Use of Cloud Storage”, July 2010
- Phishing to obtain authentication credentials
  - And other attacks involving social engineering
Attacks against the billing scheme

- Billing is a function of the usage of
  - Virtual machines/hour, traffic received/sent, CPU time consumed
- Certain attacks can cost – directly – money:
  - High number of accesses/requests/…
    - Some cloud services use automatically more resources if the usage increases (elasticity)
    - Attacker can access the service repeatedly to increase the bill of the victim (related to DDoS attacks)
- Also through the management interface
  - “Allocate 1M VMs”

Co-residence+attacks between VMs

In the cloud, virtual machines of several users can share the same physical machine (co-residence)

Attack in two steps
- The attacker instantiates several VMs until co-residence with the victim is achieved
- The attacker’s VM attacks the victim
  - e.g., using a vulnerability in the hypervisor
  - or using shared resources to obtain confidential information
SOFTWARE SECURITY IN THE CLOUD

Software

- Software is a key security problem in the cloud
  - Attacks via management interface are possible due to vulnerabilities
  - Attacks between VMs are also possible due to vulnerabilities
  - And, of course, attacks against the users’ applications (not specific in the cloud)

- A list of solutions for software security
Solution 1 – Secure programming

- Aka “do the right thing”
- Many vulnerabilities are left by programming mistakes

- Buffer overflows
  - Simply check if there is enough space in the destination buffer
- SQL injection
  - Sanitize the inputs
- Cross Site Scripting
  - Sanitize the inputs, encode the outputs

- but to err is human and code can be huge…

Solution 2 – Runtime protection

A low level example: Canaries / Stack cookies
- Compiler introduces canaries and checks

  ```
  void test(char *s) {
    push canary;
    char buf[10];
    strcpy(buf, s);
    ...
    if (canary is changed) {log; exit;};
  }
  ```

Another: Address space layout randomization.

Higher level example: webapp firewalls

<table>
<thead>
<tr>
<th>Address of buf</th>
<th>Address of s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>buf</td>
<td>canary</td>
</tr>
</tbody>
</table>
| saved ebp      | overflow     | ret address
Solution 3 – Static code analysis

- Vulnerabilities are in the source code so a solution is… to search for them
  - But it’s like finding a needle in the haystack
- Code analyzers do it automatically
  - “read” the (source) code and check if certain rules are satisfied (e.g., is memory free’d twice?)
- Commercial tools are available
  - Fortify (now HP), Coverity, Ounce Labs (now IBM)

Code analyzers work essentially in two phases
- Generate an Abstract Syntax Tree – AST (like a compiler)
- Search for vulnerabilities in the AST; several ways:
  - Syntactic analysis – check if “dangerous” functions are called (e.g., `gets` almost always vulnerable)
  - Type checking – check if data is manipulated according to its type (e.g., `unsigned int = int` is problematic)
  - Control-flow analysis – follow the control flow paths and do several checks (e.g., if there are double frees)
  - Taint checking – follow the data flow and check if input reaches dangerous functions (e.g., `strcpy`)
Solution 4 – Attack injection/fuzzing

• Look for vulnerabilities without delving into the complexity of the software, i.e., looking at it as a black box.

  - TARGET SYSTEM
  - vulnerability
  - attack
  - intrusion
  - error
  - failure

(1) Generate various attacks
(2) Look for errors / failures
(3) Find the correspondent vulnerability for that particular attack

Solution 4 – Attack injection/fuzzing

• Fuzzers
  - Late 80s/early 90s Miller/Fredrikse/So were studying the integrity of Unix command line utilities
  - During a thunderstorm one was attempting to use the utilities over a dial-up connection but the utilities were crashing
  - Data was being modified in the line due to noise
  - Thus they developed an utility called fuzz to generate random input and test the robustness of software

• Currently used to find vulnerabilities in software
  - Very successfully…
Solution 4 – Attack injection/fuzzing

- Recursive fuzzing
  - Iterating though all possible combinations of characters from an alphabet
  - Ex.: URL followed by 8 hexadecimal digits; try all possible combinations of the 8 digits

- Replacive fuzzing
  - Iterating though a set of predefined values, called fuzz vectors
  - Ex.: look for XSS vulnerabilities by providing the following inputs:
    - `&gt;&lt;script&gt;alert("XSS")&lt;/script&gt;`&
    - `"!~"&lt;XSS&gt;=&amp;{(})`

- Attack injection (AJECT project)
  - Pick a state for the target and an input to inject; put the target in that state; inject; monitor; repeat

Other solutions

- Security-aware software development processes
- Software auditing
- Testing
- Validation and encoding
- Programming language security
- Virtualization
- Trusted computing
FURTHER SECURITY: DISTRIBUTING TRUST

Security beyond software

• Some problems do not come from software (mostly)
  – Unavailability
  – Loss and corruption of data
  – Privacy/confidentiality violation – malicious insider
  – Vendor lock-in (not security)
• The malicious operator/sysadmin is particularly difficult

• Solution: distributed trust
  – Use several clouds – cloud-of-clouds
  – Each cloud has a (disjoint) set of operators
  – Assumption: there are no coalitions among clouds/operators
Example Clouds-of-clouds: DepSky

- Storage cloud formed by several storage clouds
  - Windows Azure, Amazon S3, Rackspace, Nirvanix
- Data is stored in all clouds – running a quorum algorithm
  - Any operation involves 2 steps
  - Write: 1st write metadata, 2nd write data
- Basic mechanisms
  - Data is encrypted
  - Keys are available because stored in the clouds using secret sharing
  - Cost is 2x one cloud by using erasure codes

DepSky (cont.)

- Properties
  - Availability: data is available even if one cloud is not
  - Integrity: data is not lost/corrupted even if there is a cloud failure
  - Privacy/confidentiality: data is encrypted
  - Vendor lock-in: the cost of exchanging one of the clouds is a fraction of what it might be
- Challenge: computing cloud-of-clouds
  - Data can’t be computed while encrypted
  - IaaS, running VMs
Tclouds - Trustworthy Clouds
Privacy and Resilience for Internet-scale Critical Infrastructure

• European Community project, Framework 7 (7.5 MEuro)
• Start: 1 Oct. 2010; 3 years
• Mission:
  – To develop an advanced cloud infrastructure that can deliver computing and storage that achieves a new level of security, privacy, and resilience yet is cost-efficient, simple, and scalable
  – To change the perceptions of cloud computing by demonstrating the prototype infrastructure in socially significant application areas: energy and healthcare
Conclusions

- Not an attempt to present global solutions for cloud sec.
  - Presented the main problems from the user point of view
- "Cloud computing is about gracefully losing control while maintaining accountability"
- Care with contract, analyze, monitor, security controls
- Some companies (small, medium) are probably much better with the cloud
- For others the insecurity is unacceptable
Conclusions (cont.)

• A list of solutions for software security
  – Robust coding, runtime protections, static analysis,…
• Further security: distributed trust
  – Probably needed to solve the problem of the malicious insider in the cloud
  – Plus unavailability, serious data loss, vendor lock-in
• Research is needed

Thank you. Questions?

• Blog - http://www.seguranca-informatica.net/
• Book - http://seguranconosoftware.blogspot.com/
• TCLOUDS - http://www.tclouds-project.eu/