Security design in the cloud

Security in software/applications

- Legacy software provided security features for the buyer to configure for securing the application
  - Vendors make it easy to integrate with enterprise security data stores
  - Active Directory and SSO
  - Run within buyer’s security parameter behind a corporate firewall
- Cloud computing gives more responsibility to CSP to secure software on behalf of consumer
  - Consumer gives up control and allows data to live outside firewall
  - Onus on CSP to comply with various regulations and security standards
- Myth: Critical data in the cloud cannot be secure
  - Security must be architected into system regardless of data location

Data in the cloud

- Regulations, such as HIPAA, do not dictate the location of data
  - They dictate that personally identifiable information, such as demographic data, credit card numbers, or health-related information, must be encrypted at all times
- Government’s ability to seize data from CSP
  - Patriot Act
    * Require any US-based company to fulfill request for data from any of its data center, even outside the US
    * Data can be requested regardless of whether it is in the cloud
    * If the data is encrypted, company will have to be notified and asked to decrypt it
  - Company may be asked to provide data even if data is not in the cloud

Extent of security required

- Target industry
  - Determines the scope of regulations
    * High level of security for health care, government, or financial industry
    * Moderate security for social web and online gaming
    * Stringent security for B2B services, with a minimum base line for third party vendors
  - Consumer-facing or business-to-consumer services offer use-at-your-own-risk services with some privacy but little about security and regulations
    * Facebook puts the onus of user account security on user
- Customer expectations
  - Does the customer want to be in cloud?
    * May have to choose hybrid cloud to satisfy the customer
- Sensitivity of data being stored
Security Design in the cloud

– Public vs private information
– Encrypt data at rest

• Risk tolerance
  – Companies may be more security conscious for image than required by regulation
  – Speed to market vs security controls

• Maturity of product
  – Product may not need stringent security initially but may need it as it matures

• Transmission boundaries
  – Source and destination of data
  – Data transmission within the data center or crossing international boundaries

• Other factors
  – Actors involved (build vs buy)
  – Criterion to meet security requirements
  – Timeline and deadlines

Responsibility for each cloud service model

• Further up the cloud stack, more security responsibility for CSP

• Four categories in cloud stack
  1. Infrastructure layer
     – Physical things such as data centers, servers, hardware and peripherals, network infrastructure, storage devices
     – Provide security for all physical infrastructure, if using private cloud
     – CSP to provide security for infrastructure in case of public cloud
     – CSP typically invests a large amount of time, money, and human capital to build top-class security and should be able to provide certifications for different regulations

  2. Platform layer
     – CSP secures the underlying application software layer, including OS, application servers, database software, and programming languages
     – CSP also manages other application stack tools for on-demand services such as caching, queuing, messaging, e-mail, logging, monitoring (managed by consumer in IaaS)
     – When the PaaS provider leverages the services of an IaaS provider, such as Heroku and Engine Yard using AWS, PaaS provider secures the application stack while IaaS provider secures the underlying infrastructure
     – Public-unmanaged deployment model
       * PaaS provider deploys on an IaaS provider’s public cloud
       * Consumer takes on the responsibility of managing and patching both PaaS software and application stack
       * Common for hybrid cloud
     – Private-hosted model
       * A private PaaS is deployed on an externally-hosted private IaaS cloud
       * Infrastructure handled by IaaS provider
       * Consumer manages and secures the application stack and security
     – Private-managed model
       * IaaS cloud is private, either externally hosted or within the consumer’s own data center
Security Design in the cloud

* External IaaS provider
  - Consumer hires a service provider to secure and manage PaaS and application stack
  - IaaS provider manages and secures infrastructure layer
* Internal IaaS cloud
  - Consumer manages and secures infrastructure layer
  - Managed service provider manages and secures PaaS layer

- Private-unmanaged model
  * Consumer responsible to secure entire cloud stack plus the PaaS software
  * Private IaaS plus managing PaaS solution in data center
  * Keep data out of public cloud

3. Application layer (SaaS)

  - Application development focus on things such as secure transmission protocols, data encryption, authenticating and authorizing users, and protecting against web vulnerabilities
  - Application security provided by CSP

4. User layer

  - Consumer performs user administration

Security strategies

• Typical cloud architecture
  - Applications built as distributed to scale horizontally with increase in demand
  - Architecture may contain a dedicated web server farm, web service farm, caching server farm, and database server farm
  * Redundant farms across multiple data centers

• Key strategies to manage security in a cloud-based application

1. Centralization
  - Practice of consolidating a set of security controls, processes, policies, and services
  - Reduce the number of places to manage and implement security
  - Single sign on instead of authenticating for individual application

2. Standardization
  - Security as a core service to be shared across the enterprise instead of a specific application
  - Implement industry standards for accessing systems (OAuth or OpenID) to connect to third parties
  - Application of standardization
    (a) Industry best practices to implement security solutions and selecting encryption method, authorization, and API tokenization
    (b) Implement security as a stand-alone set of services shared across applications
    (c) Standard naming conventions and formats for outputs such as logs, errors, warnings, and debugging data

3. Automation
  - Automate repeatable steps like creating environments and deploying software
  - Implement proper security controls and processes within automation scripts
  - Helps with scaling as well

Areas of focus
• PDP strategy
  1. Protection
     – Security controls, policies, and processes to protect the system and company from security breaches
  2. Detection
     – Process of mining logs, triggering events, and proactively trying to find vulnerabilities within the system
  3. Prevention
     – Take necessary steps to prevent damage

• Policy enforcement
  – Rules to manage security within a system
  – Rules should be configurable and decoupled from the applications
  – Policies maintained at every layer of cloud stack
     * At user layer, policies maintained in a central data store like active directory
     * User information maintained and accessed through protocols such as LDAP
  – Changes to user data and rules managed in central data store and not within application
     * Exception: Rules specific to application are managed within application
  – Application layer
     * Maintain application-specific rules in centralized data store abstracted from actual application
     * Central data store can be managed by database, XML file, or a registry, and accessed via API
  – Application stack layer
     * OS, database, application server, development languages
     * In IaaS, enforcement accomplished by scripting infrastructure provisioning process
     * Template for each unique machine image (gold image) containing security policies around access, port management, and encryption
     * Use gold image to build cloud servers to run application stack and applications
     * Update gold image to reflect change in policies
     * Automate entire process to eliminate human error

• Encryption
  – Always encrypt sensitive data in cloud, whether at rest or during transmission
  – Entails overhead
  – Personally identifiable information (PII)
     * Demographics (name, social security number, address)
     * Health information (biometrics, medications, medical history)
     * Financial information (credit card number, bank account number)
  – System information that can aid in attacks
     * IP addresses
     * Server names
     * Passwords
     * Keys
  – Sensitive data can be encrypted at attribute level, row level, or table level
     * Simplicity – Manage encryption at table level
     * Performance – Frequency of data access
     * Traceability – Isolate attributes in scope and track all API calls
Security Design in the cloud

- **Key management**
  - Public and private cryptographic key pairs
  - Object encrypted by public key and unencrypted by private key
  - Do not store keys as clear text
  - Do not refer to key directly in the code
  - Keys stored outside application and accessed by a single secure method
  - Rotate keys at regular frequency
    - Mitigate risk if keys are compromised
    - What if an employee with access to keys leaves the company
  - Never store keys on the same server that they are protecting

- **Web security**
  - Hacking into web servers
    - Intercept data in transit, inject SQL statements, hijack user sessions, and other malicious behaviors
  - Leverage frameworks, such as .NET
  - Use a web vulnerability scanning service
    - There are SaaS solutions for these types of scans

- **API management**
  - Representational State Transfer (RESTful) web APIs
  - Integrate different cloud services by leveraging APIs
  - Standards to share APIs with partners and customers
    - Support for OAuth and OpenID
    - Basic authentication over SSL
  - Do not use passwords but use API keys between client and provider

- **Patch management**

- **Logging, monitoring, and auditing**

**Centralized logging strategy**

**Logging**

- Critical component of any cloud-based application
- Systems are built to scale up or down on demand
  - Compute resources are provisioned and deprovisioned with workload spikes
- Information must be preserved when the cloud resources go away
  - Separate the storage of logging information from the physical server that creates the log

**Log file uses**

- Information on behavior of various systems
  - Database activity, user access, errors, debugging information
Security Design in the cloud

– Centralized logging helps in discovering anomalies over a number of servers

• Troubleshooting
  – Collect debugging information and error messages in production environment

• Security
  – Track all user access, successful and unsuccessful
  – Intrusion detection and fraud detection

• Auditing
  – Trail of data
  – Documentation of processes and controls may not be enough
  – Documents need to be backed up with real data from logs
  – Logs maintained on a separate server farms
    * Helps in removing developer access from all production servers

• Monitoring
  – Identify trends, anomalies, thresholds, and other variables
  – Resolve issues before they become noticeable and impact end users

Logging requirements

1. Direct logs to an isolated storage area
   • Design for centralized logging solution in an IaaS service

<table>
<thead>
<tr>
<th>Production Servers</th>
<th>Admins (total access)</th>
<th>Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Servers</td>
<td></td>
<td>SYSLOG</td>
</tr>
<tr>
<td>DB logs, App server logs, Web logs</td>
<td></td>
<td>Logging Service</td>
</tr>
<tr>
<td>API Servers</td>
<td></td>
<td>SYSLOG</td>
</tr>
<tr>
<td>DB logs, App server logs, Web logs</td>
<td></td>
<td>Logging Service</td>
</tr>
<tr>
<td>Database Servers</td>
<td></td>
<td>SYSLOG</td>
</tr>
<tr>
<td>DB logs, App server logs, Web logs</td>
<td></td>
<td>Logging Service</td>
</tr>
<tr>
<td>Utility Servers</td>
<td></td>
<td>SYSLOG</td>
</tr>
<tr>
<td>DB logs, App server logs, Web logs</td>
<td></td>
<td>Logging Service</td>
</tr>
</tbody>
</table>

– All logs directed to syslog instead of local machine
– Syslog piped directly to a dedicated logging server farm
– Logging server farm redundant across data centers, or availability zones
– Logging data transformed into a NoSQL database using some logging solution (open source or commercial)
– Tools allow to search logs, schedule jobs, trigger alerts, and create reports
– Benefits
  * Block developer access from all servers in production environment
    · Developers access only logging servers
    · Access through a secure user interface or through API
  * Simpler audit because of centralized logs location
  * Allows data mining and trend analysis because of NoSQL database
  * Simpler intrusion detection – tools run on top of central logging database
  * Loss of data is minimized
• Data not stored on local disk of servers that may be deprovisioned on the fly

• Leverage SaaS logging solution
  – Logs sent to a cloud-based centralized Database as a Service solution
  – No need to build, manage, and maintain logging functionality
  – Logs maintained off-site on a scalable and reliable third-party cloud infrastructure
  – Logging service not impacted by any part of the data center going down
  – Log files from different CSPs can be managed and maintained in one place

• PaaS solutions
  – May be integrated with a logging SaaS solution
    * Logging add-on or plug-in for PaaS
    * PaaS user does not have to write code or figure out integration with SaaS solution
  – SaaS solution provides access to centralized logging solution

2. Standardize log formats

• Standardize log formats, naming conventions, severity levels, and error codes for all messages

• Utility service to write application messages with a common log message format
  – Design APIs to use standard http error codes
  – Leverage a standard for severity levels

<table>
<thead>
<tr>
<th>RFC 5424 Severity Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

• Common vocabulary for error descriptions
  – Tracking attributes such as date, time, server, module, API name
  – Helps with search from logging tools for consistent results
  – Use a database or XML data store to pull messages
    * Also enforces consistency when things need to be changed

• Standardization
  – Help optimize search and produce consistent results
  – Helps detect patterns and alert appropriate personnel in case of inconsistency in log results
  – Automatic generation of audit reports
  – Trend reports can be derived, detecting common issues
  – Statistics tied to deployments to proactively analyze the quality of each deployment

SLA Management

Service Level Agreement

• Agreement between CSP and cloud service consumer (CSC)
  – Sets expectation of the level of service promised by CSP to CSC
  – Critical for cloud-based services because CSPs take on responsibilities on behalf of CSC
– Consumer can expect services that are reliable, scalable, secure, and available

- Company building cloud services on top of IaaS or PaaS provider has to consider SLA of its CSPs
- Company needs to establish SLAs to satisfy the needs of its customer base

**Factors that impact SLAs**

- Companies may use different vendors to build different parts of an application
  - IaaS CSP for infrastructure layer
  - PaaS provider for application stack layer
  - Collection of SaaS solutions and third-party APIs for core utility functions

- Each CSP may have his own SLA

- Customer expectations
  - First step in defining an SLA
  - Influenced by customer characteristics, criticality of services provided, and types of interactions between provider and consumer
    * Consumer vs enterprise customer
    * Paying vs nonpaying customer
    * Regulated vs non-regulated industry
    * Anonymous vs personally identifiable
  - Performance, uptime, reliability
    * Terms of service weighted towards protecting CSP
    * Offer the services on an "as is" basis
    * Consumers must accept the services to participate
    * CSP may promise to make best effort to secure consumer data and maintain privacy
  - Strength of SLA
    * Nonpaying consumers may be offered lower SLAs than paying consumer
    * Free trial services
    * Regulated industry – performance, uptime, security, privacy, compliance
    * Social media – just privacy
    * High levels of security and privacy if any PHI is collected
    * Critical services (social media vs retail services)

- Different SLA for different parts of products

- Inventory of all actors involved in providing and consuming cloud services
  - SLA and track record of CSPs
  - Devise strategy in the event of an outage
  - Fail-over plan if CSP fails
  - Level of service (IaaS/PaaS/SaaS) and availability/control

**Defining SLAs**

- Company could be both a cloud consumer and cloud provider
– SLAs for companies that build on public, private, or hybrid clouds

• Metrics-based SLAs
  – Overall uptime of application/service
  – Page-load times
  – Transaction processing times
  – API response times
  – Reporting response times
  – Incident resolution times
  – Incident notification times

• Tracking and reporting
  – Accomplished through a combination of logging and monitoring

• Regulatory, security, and privacy perspective
  – Security and privacy safeguards
  – Published incident response plan
  – Web vulnerability scans and reports
  – Published disaster recovery plans
  – Safe harbor agreement
  – Backup and recovery process documentation
  – Source code escrow

• Periodic (monthly) reporting of metric-based SLAs
  – One document to summarize all security, privacy, and regulatory controls in place and provide to customer on request
  – Certifications, audit reports, web vulnerability scan report, monthly metrics report

• Building secure web applications
  – Leverage web frameworks and keep them current
    – Django for Python, .NET for Microsoft, Zend for PHP, Ruby on Rails, Struts for Java

• Automation and policy enforcement

Managing vendor SLAs

• 99.9 to 100 percent uptime for well-established IaaS and PaaS providers

• No published SLA for top SaaS solutions such as Salesforce.com and Concur

• Handling customer issues, such as outages
  – Refund or credit
  – Does not repair collateral damage to customer business

• Service levels represent the uptime of infrastructure (for IaaS)
  – Developers need to build highly available applications on top of the infrastructure
– Build for redundancy across multiple zones on AWS

**Monitoring Strategies**

**Monitoring**

- Process of testing and verification of application performance as expected
- High level of uptime, reliability, and scalability in cloud services

**Proactive vs reactive monitoring**

- Reactive monitoring
  - Monitoring typically used to detect failures
    * Consumption of memory, CPU, and disk space of servers
    * Throughput of network
    * Tools to ping URLs to check the availability of websites
  - Focuses on detection
- Proactive monitoring
  - Focuses on prevention
  - Define healthy system metrics
  - Watch patterns to detect if data trends towards an unhealthy system
  - Fix the problem before reactive monitors cause an alert

**What needs to be monitored?**

- SLAs define a contract between CSP and consumer
- Each SLA must be monitored, measured, and reported on
  - Metric-based SLAs – response time, uptime
  - SLAs focusing on processes around privacy, security, and regulations
- Multiple parts of a cloud-based system
  - Any part of the system can be a point of failure and needs to be monitored
  - Role-based information need
    * Front-end developer concerned with page-load times, network performance, and performance of APIs
    * DBA concerned with database server metrics in threads, cache, memory, CPU utilization, and SQL statement response times
    * Sys admin concerned with requests per second, disk space capacity, and CPU and memory utilization
    * Product owner concerned with visits per day, new users, cost per user, and business-related metrics
- Technical vs business metrics
  - Decline in customer use of apps
  - Assessing the success of each deployment
  - Use of various options in the app
Monitoring strategies by category

• Performance
  – An important metric within each layer of the cloud stack
  – User layer
    * Track how users interact with the system
    * Number of new customers
    * Number of unique visitors per day
    * Number of page visits per day
    * Average time spent on site
    * Revenue per customer
    * Bounce rate (percent of users who leave without viewing pages)
    * Conversion rate (percent of users who perform desired action based on direct marketing)
  – User characteristics based on a system
    * Number of new users
    * Number of unique users per day
    * Number of calls per user per day
    * Average time per call
    * Revenue per user
  – Application layer
    * Measurement of response of system to end user
    * Common metrics
      · Page-load times
      · Uptime
      · Response time (APIs, reports, queries)
    * Tracked and aggregated at different levels
    * Components of system to be tracked independently due to unique performance requirements and SLAs
      · Consumer-facing web page
      · Collection of APIs
      · Administrator portal for data management
      · Reporting subsystem
  – Application stack layer
    * Similar to application layer
    * Track the performance of underlying components of application stack
      · OS, application server, database server, caching layer
    * Every component monitored on each machine
    * Servers need to be monitored in clusters/groups to compute the metrics for a given customer
  – Infrastructure layer
    * Metrics apply to physical infrastructure such as servers and network routers
    * IaaS provider may host a web page to show the system health
      · Typically limited to functioning-normally, having-issues, or completely-down

• Throughput
  – Measurement of average rate at which data moves through the system
  – User layer
Security Design in the cloud

- Number of concurrent users or sessions processed by system
  - Application layer
    * Amount of data transmitted from application stack layer through the application layer to the end user
    * Transactions per second, requests per second, click through per second, or page visits per second
  - Application stack layer
    * Critical in diagnosing issues within the system
    * Tools or SaaS solutions to set alerts and notifications when certain thresholds are met
    * Provide analytics to spot trends in the data
  - Infrastructure layer
    * Measurement of flow from physical servers and network devices

- Quality
  - Measurement of accuracy of information and impact of defects on the end user in a production environment
  - Should focus on
    * Accuracy
    * Correctness of data being returned to end users
    * Error rates
    * Frequency of errors
    * Deployment failure rates
    * Percentage of time deployments fail
    * Customer satisfaction
    * Perception of quality and service
  - Standardization of data collection
    * Error codes, severity levels, log record formats
    * Common error and logging APIs
    * Consistent data to centralized logging system
    * Automated reports and dashboards to mine the data from logging systems
      - Relevant key metrics including quality, error rates, and error types
    * Thresholds for alerts and notifications
  - User layer
    * Success and accuracy of user registration and access
    * Quality could be a usability issue instead of a defect
      - Cumbersome or confusing user interface
  - Application layer
    * Defect types
    * Errors related to erroneous data, failed transactions, 400- to 500-level http response codes
    * Track errors for each API and for each module within the system
  - Application stack layer and infrastructure layer
    * Log and track errors for each component

- KPIs – Key performance indicators
  - Inform if the system is meeting business goals
  - Unique to each company’s business model
  - Help proactively detect potential issues
    * Revenue per customer
Security Design in the cloud

* Revenue per hour
* Number of incoming customer calls per day
* Number of jobs completed per day
* Site traffic
* Shopping cart abandonment rate
  - Typically measured at the application layer

• Security
  - Threat landscape keeps on changing
  - Proactively monitor all components for suspicious patterns
  - Monitor for security at every layer of cloud stack
    * Authentication for access to each layer
    * Check for authentication failure at each level and see if it happens for a certain user/system/IP address
    * Bot attacks
  - Blacklist the offending IP address
    * Check how the intruder gain access in the first place
    * Security audits

• Compliance
  - Alerts when something falls out of compliance
  - Policies and procedures must be followed within the system and within the business
    * Background check on employees
    * Restricted access to buildings
    * Access on a need-to-know basis

Monitoring by cloud service level

• SaaS
  - Little can be done by end user
  - Service is either up or down
  - Possible to have a second SaaS solution in case the primary one goes down

• PaaS
  - Public PaaS
    * Vendor manages both infrastructure and application stack layer
    * PaaS vendor supplies APIs to various monitoring and logging solutions for integration
    * All logs go to PaaS-provided central logging system
    * Vendor owns responsibility for security and consumer focuses on application
  - Private PaaS
    * More like IaaS
    * Consumer must monitor the system down to application stack layer
    * Plug-ins for logging and monitoring solutions

• IaaS
  - Logging and monitoring installed and managed by consumer