Pragmatic Cloud Security Patterns

Rich Mogull
@rmogull
Little. Cloudy. Different.

- Cloud can be more secure than traditional datacenters.
  - The economics are in your favor.
  - Cloud architectures can wipe out some traditional security headaches.
- This isn’t theory, it’s being done today.
  - But only if you understand how to leverage the cloud.
- We will show you how.

*Embrace* the cloud and *extend* your program.
Building the Foundation

• Focus on core security patterns
  • Architecture
  • IAM
  • Monitoring/alerting
  • Netsec
  • VA | Server/instance/container | change management | IR

• Leverage cloud characteristics for better security
  • Most examples will be AWS, but principles apply everywhere
For clients to use a cloud provider, they must trust the provider.

This is especially true for anything with a sensitive data or process.

Thus security has to be a top priority for a provider or you won’t use them.

A major breach for a provider that affects multiple customers is an existential event.
Cloud Provider Critical Security Capabilities

- API/admin activity logging
- Elasticity and autoscaling
- APIs for all security features
- Granular entitlements
- Good SAML support
- Multiple accounts per customer
- Software defined networking
- Region/location control
- *Nice to have*: infrastructure templating/automation
Provider and Account Segregation
Segregation is critical but hard

- Segregating networks in a data center is hard, expensive, and often unwieldy.
- It’s hard to isolate application services on physical machines.
  - Even using virtual machines has a lot of management overhead.
- Attackers drop in and move North/South in application stacks, and East/West on networks (or both).
Limiting blast radius
To a host or network...
To a host or network…
Or an entire “data center”
Or an entire “data center”
Traditional blast radius
<table>
<thead>
<tr>
<th><strong>Characteristics</strong></th>
<th><strong>AWS</strong></th>
<th><strong>Azure</strong></th>
<th><strong>GCP</strong></th>
<th><strong>OpenStack/VMware</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest/Oldest/Most Services</td>
<td>Excellent MS stack support. Many services. Growing *nix support</td>
<td>Newer, wide range of services. Cost competitive. Great container support.</td>
<td></td>
<td>Both struggle for different reasons. Very expensive to run at scale.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Security</strong></th>
<th><strong>AWS</strong></th>
<th><strong>Azure</strong></th>
<th><strong>GCP</strong></th>
<th><strong>OpenStack/VMware</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>All critical capabilities</td>
<td>Limited API auditing, dedicated security center</td>
<td>Beta auditing. Variable IAM granularity</td>
<td></td>
<td>Varies based on deployment. OpenStack especially.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Account Structures</strong></th>
<th><strong>AWS</strong></th>
<th><strong>Azure</strong></th>
<th><strong>GCP</strong></th>
<th><strong>OpenStack/VMware</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization &gt; Accounts &gt; Regions</td>
<td></td>
<td>Azure AD &gt; Subscriptions</td>
<td>Organization &gt; Projects</td>
<td>Tenants</td>
</tr>
</tbody>
</table>
Baseline Security Infrastructure and Operations
Management Plane IAM

• Separate entitlement matrix per-project and account.
• Map roles to rights based on matrix, don’t just default.
• Brokers can be very useful.
• Keep code identities in the cloud.
IAM Best Practices

- Federate everything; ideally with one authoritative source
- Leverage within-provider capabilities (e.g. Cognito)
- Use a cloud-based directory for in-cloud customer/employee access
- Always have a good-old out-of-band back door admin account
- Isolate. Isolate. ISOLATE.
- MFA all the things
Monitoring and Alerting Differences

Management Plane

Velocity

Distribution/Segregation

Account

Virtual Network

Subnet

Security Group

Account

Virtual Network

Subnet

Security Group
Management Plane Security Monitoring

Direct

Proxy
Cloud Provider Support

- Nearly all API calls (CloudTrail)
- Configuration state over time (Config)
- System logs (CloudWatch)
- Threat Intel (Guard Dog)
- Continuous vuln assessment (Inspector)

- System/generic sources (Log Analytics)
- Configuration state + security incidents (Azure Security Center)
- Direct activity/API collection in console

- Partial API logging (Compute/App Engine, BigQuery, SQL/VPN/Storage, Deployment Manager)
- Central log service (Stackdriver)
- Continuous Assessment (Cloud Security Scanner)
Cascading Architecture

- Dev
- Test
- Prod

Logs

Project Accounts

Security Account

Security Monitoring

SIEM In Datacenter
Even Driven Alerting
Built-in Options

- Push to CloudWatch logs via a connector agent
- Push to Log Analytics using agent and integrate with Azure Security Center
- Extensive StackDriver support
Build Your Own (or PaaS)

- Multiple vendors and open source projects integrate with major cloud providers
  - Some as SaaS/PaaS
  - Others as virtual machines on the marketplace
  - Or software you install yourself
- Use object storage as much as possible to save costs and add resiliency
- Cascading architectures very important
Cascading System Logs
Network segregation by default

Granularity of a host firewall with the manageability of a network firewall
**Azure and GCP**

- Virtual networks are similar to VPC
- By default, Network Security Groups are open to the same subnet
- NSGs apply to subnets and to instances
- NSGs use an ACL-like structure with prioritization, which is very different than AWS
- Azure also supports fully public services that use a different mechanism (endpoints)

- Firewalls are default deny- no rules, no access
- Inbound rules only
- If a rule isn’t defined, the packet is dropped, both internal and external
- In some ways closer to AWS, but not as robust
NetSec Rules of the Road

• **Use architecture, not boxes!**
• There is no DMZ
• Trust security groups, they are better than your firewalls
• Avoid virtual appliances as much as possible
  • Use agent-based to fill gaps, if you can
• Account for serverless
• Isolate with virtual networks
• Allow more software VPNs
• Get used to not seeing all the packets… it’s a security benefit
Cloud “DMZ”
Network attack path?
Hybrid Networks
<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immutable</td>
<td>- Based on images and automatically deployed (e.g. by an auto scale group)</td>
</tr>
<tr>
<td></td>
<td>- Login disabled since changes won't propagate to other instances</td>
</tr>
<tr>
<td></td>
<td>- You replace with new versions instead of patching/updating old versions</td>
</tr>
<tr>
<td></td>
<td>- Very easy to harden for security (e.g. disable SSH)</td>
</tr>
<tr>
<td>Automated Configuration</td>
<td>- The virtual machine is automatically configured using a template/policy</td>
</tr>
<tr>
<td></td>
<td>based tool (e.g. Chef / Puppet / Ansible / Salt)</td>
</tr>
<tr>
<td></td>
<td>- It changes, but manual changes disabled since the automation would</td>
</tr>
<tr>
<td></td>
<td>overwrite</td>
</tr>
<tr>
<td>Standard / Long Running</td>
<td>- Managed just like traditional servers</td>
</tr>
</tbody>
</table>
Vulnerability Assessment Options

- Trad/Network
- Assessment Agents
- Deployment Pipeline
Incident Response

- Cloud registry
- Who to call and out of band authentication
- Cloud IR toolkit- centralized automation is ESSENTIAL!
- Focus on the management plane first, servers/containers second (or third, or fourth)
- Lock down IAM before anything else
- Multiple accounts dramatically improve IR success rates
- Did I mention automation? Quarantine environments?
Level Up with DevOps, Architecture, and Automation
CI/CD Pipelines for Security

- Source Code
- Cloudformation Templates
- Chef Recipes

Functionality Tests
- Non-Functional Tests
- Security Tests

Jenkins

Test → Prod
Pattern 3: Data Transfer
PaaS Air Gap

No direct network connection
Platform Architecture

- Collections of scripts/code, not a monolithic application.
- Framework for:
  - Auth
  - Scheduled activities
  - Logging
Self-Healing Infrastructure (yes, for real)

Change a security group

Event Recorded to CloudTrail

Passed to CloudWatch Log Stream

Lambda Function analyzes and reverses

Triggers an CloudWatch Event
- Configure IAM Roles
- Create S3 bucket
- Set bucket permissions
- Create CloudWatch Log Group
- Enable CloudTrail
- Configure CloudTrail/CloudWatch Log connection
- Create CloudWatch Alarm
- Create SNS notification topic

Account 123

SecServer
Role: Sec

Account 456
Getting Started

- Account architectures and IAM are the key to a good start
  - Isolate with accounts; it looks harder up front but avoids cloud technical debt that is very hard to unwind after the fact
- Embrace cloud technologies
- Extend your existing programs and controls
- Start with a few smaller projects to get practice and build out
- We’ve barely scratched the surface, and CLOUD PROVIDER MATTERS!
- Be a cloud native, not a tourist. Everyone hates tourists