Security as Code: The Time is Now

Dave Shackleford
Founder, Voodoo Security
Sr. Instructor, SANS
Introduction

• Business is moving faster to the cloud, and DevOps is accelerating scale and pushing automation
• Where's infosec? How do we secure DevOps and cloud deployments?
• Security needs to change how we work with operations and the business
• DevSecOps is one way to better automate and integrate security for the cloud
This was your data center before...
This is your data center soon...
What is DevOps?

• An open dialogue and collaboration between development and operations teams
  – The goal is often “continuous integration” and/or “continuous delivery”

• DevOps goals:
  – Automated provisioning
  – No-downtime deployments
  – Monitoring
  – “Fail fast and often”
  – Automated builds & testing
DevSecOps Integrates Security

• DevSecOps strives to automate core security tasks by embedding security controls and processes into the DevOps workflow
• Originally focused primarily on automating code security and testing
  – Primarily code analysis, unit tests
• Now also encompasses more operations-centric controls
  – Logging and event monitoring
  – Configuration and patch management
  – User and privilege management,
  – Vulnerability assessment
Cloud Security: Where to Focus

Internal:
- Deployment pipeline
- Data security

External:
- CSP security controls
- Integrated controls/monitoring

AMI is available

Instances

VPC Subnet

Availability Zone

Router

Virtual Private Gateway

VPN Connection

Customer Gateway

Customer Network
General: Start with Cloud Threat Modeling

- Most likely threats
- Data types and sensitivity
- System builds and controls
- Cloud environment security posture
- Existing controls in place
- Controls we “lose” in the cloud
## Map to Cloud Risk Considerations

<table>
<thead>
<tr>
<th>Security Considerations</th>
<th>Cloud Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SaaS</td>
</tr>
<tr>
<td>Virtual network security</td>
<td></td>
</tr>
<tr>
<td>VM instance template management</td>
<td></td>
</tr>
<tr>
<td>System build configuration</td>
<td></td>
</tr>
<tr>
<td>Antimalware</td>
<td>X</td>
</tr>
<tr>
<td>Data security at rest and in transit</td>
<td>X</td>
</tr>
<tr>
<td>Administrative console security</td>
<td>X</td>
</tr>
<tr>
<td>Roles and privileges</td>
<td>X</td>
</tr>
<tr>
<td>Logs and monitoring for activity</td>
<td>X</td>
</tr>
<tr>
<td>Sensitive data and policy compliance (DLP)</td>
<td>X</td>
</tr>
</tbody>
</table>
Where does our traditional model fail?

• Perimeter focused
• Rely on dedicated appliances
• Heavy footprints for endpoints
• Strict change controls and schedules
• Slow rate of change (again…schedules)
• Lack of automation
• No detection of lateral movement
Security as Code?

• With DevOps and “Infrastructure as Code”, we define everything in a software-defined method:
  – Servers (usually VMs)
  – Containers
  – Application stacks
  – Networks
  – Roles/Privileges/Access models

• Security needs to be defined in this way, as well
Deployment Pipeline Security

• Focus on:
  – Code security
  – Code repositories
  – Automation tools
  – Orchestration platforms
  – Gateways and network connectivity

• Authentication/Authorization and privileged user monitoring and management are critical
Development/Deployment Integration

- We need to integrate into deployment pipelines
- Continuous Integration vs. Continuous Deployment
- Early: Static and Dynamic code analysis
- Early: Defined libraries and configs
- Later: Monitoring and Control in instances
Specific Control Examples

• Application-level security through CI/CD integration
  – SAST (Veracode or Fortify on Demand is an example)
    • Trigger automated build upload to Fortify with Jenkins
  – DAST
    • Trigger automated WebInspect or AppScan scan

• Deployment Infrastructure
  – Automation/Orchestration tool lockdown
  – Roles/Privileges/Keys – Ansible Vault or Tower

• Infrastructure-level security
  – Configuration and hardening via Ansible or Puppet
  – Docker security verification during CI/CD build
Ansible Example: RHEL 7 STIG

- name: "HIGH | RHEL-07-010440 | PATCH | The operating system must not allow empty passwords for SSH logon to the system."

  lineinfile:
    dest: /etc/ssh/sshd_config
    regexp: (?i)permitemptypassword
    line: PermitEmptyPasswords no
    validate: sshd -t -f %s
  notify: restart ssh

tags:
  - cat1
  - high
  - patch
  - RHEL-07-010440
  - ssh
The key is a feedback loop

- For true DevSecOps to take hold, security teams will need to embed automated tests and validation of controls into the deployment cycle.
- Monitor continuously in production with “triggered” responses that can roll controls back to a known good state.
DevSecOps and Configuration State

- Define configuration items and baselines
- Approve configuration templates and controls
- Embed configuration standards in builds and automate patch management
- Monitor everything!
- Roll back if a “diff check” fails
Vulnerability scanning

• Continuous monitoring is critical to DevSecOps success
• Check for scanning products that have been adapted to cloud
  • Some have strong API support and integration
• Also consider host-based assessment
  • This can be automatically triggered in deployment and monitoring environments
Security as code: Privilege management

- Carefully limit and control the accounts and privileges assigned to resources
- All users, groups, roles, and privileges should be carefully discussed and designated to resources on a “need to know” basis
- Assign “least privilege” and monitor carefully
- Embedded tokens and keys in code are common
Collect and analyze logs and events

• Logs and events generated by services, applications, and operating systems within cloud instances should be automatically collected

• Organizations implementing DevSecOps should:
  • Collect the appropriate logs
  • Send logs to secure central logging services or cloud-based event management platforms
  • Monitor events closely using SIEM and/or analytics tools
Security as Code: Define Policies

• Define policies for components, networks, and more
• This might include:
  – Configurations (Puppet, Chef)
  – App deployment and automation (Ansible, Jenkins)
  – Additional orchestration and automation tools
• Cloud providers may offer tools, too (CloudFormation in AWS, for example)
Security as Code: Define security “stories”

- These will be specific use cases and requirements:
  - Input validation for app X
  - Use of TLS for all communications
  - Hardening to CIS Benchmark standards
- These are then implemented IN code and vetted, or via policy files and language
Security as Code: Internal Build and Deployment Security

• For the internal side of Security as Code, imagine the following:
  – Automated code scans upon check-in
  – Automated app scanning in test/staging
  – Automated Server, Container, and Network configuration checks via policy
  – Continuous monitoring of all core components in the Deployment Pipeline
Security as Code: Test policies regularly

• Using build testing tools like Test Kitchen and Vagrant can simplify internal policy validation

• Coordinate penetration tests and routine checks to validate policies’ effectiveness
  – Are only required ports open?
  – Are credentials secured?
  – Are encryption keys secured?
  – Are privileges assigned properly?
Security as Code: Automate Production Feedback Loops

• That whole “continuous monitoring” thing?
  – Yeah, it’s time.

• You need detection and response playbooks, too:
  – Scheduled checks of X generates alert/log
  – Alert triggers automated process Y

• All of this needs to be automated
  – Some critical tasks may require a human sign-off
What does this all mean?

- We have HUGE gaps in security involvement and knowledge for all of this
  - Cloud-oriented threat modeling
  - Understanding of DevOps processes and tools
  - Ability to write roles or playbooks for Ansible and other platforms
  - Understanding of software-defined tools, APIs, and integration capabilities
It’s time to shift…

• From THIS:

• To THIS:

Type: "AWS::EC2::SecurityGroupIngress"
Properties:
  CidrIp: String
  CidrIpv6: String
  FromPort: Integer
  GroupId: String
  GroupName: String
  IpProtocol: String
  SourceSecurityGroupName: String
  SourceSecurityGroupOwnerId: String
  ToPort: Integer
Integrate DevSecOps into security operations

• Leverage automation:
  • Salt
  • Puppet
  • Chef
• Embedded agents and SecaaS options
• Consider:
  • Defensible infrastructure
  • Operational discipline
  • Situational awareness
  • Countermeasures
DevSecOps: A Checklist

- Ensure that periodic reviews of the overall risk posture within cloud environments are performed
- Keep system instances in the cloud as locked down as you can
- Pay careful attention to privilege allocation and user, group, and role management.
- Commit to a culture of continuous monitoring
- Discuss vulnerabilities detected in cloud deployments with all team members
- Ensure you are gathering adequate security and operations logs and event data, sending this to a remote monitoring and collection platform.
- Discuss the changing threat landscape with DevOps teams
Questions?

Dave Shackleford
@daveshackleford
daveshackleford.com
voodoosec.com