Aligning AWS Services with Zero Trust Pillars

The Zero-Trust model is founded on the principle of "never trust, always verify" for every request,
regardless of where it originates. Below is a detailed alignment of AWS services with each of the
Zero-Trust pillars, followed by practical testing steps to validate Zero-Trust principles using AWS resources.

1. Pillar: Identity and Access Management (IAM)

Alignment with AWS Components:
• AWS Identity and Access Management (IAM): Manage users, roles, and policies to define fine-grained access controls.
• AWS IAM Identity Center (SSO): Provide centralized authentication and single sign-on for workforce users.
• Amazon Cognito: Manage application users, implement MFA, and issue JWT tokens for authentication.
• Enforce Multi-Factor Authentication (MFA) and use IAM Condition Keys for context-based access controls.
• Use AWS Verified Access for device and identity-based access to internal applications.

2. Pillar: Network Segmentation

Alignment with AWS Components:
• Amazon Virtual Private Cloud (VPC) and Subnets: Segment workloads across public, private, and database subnets.
• Security Groups (SGs) and Network ACLs: Enforce traffic rules at the subnet and instance levels.
• AWS Network Firewall: Provide centralized traffic inspection and filtering between network segments.
• AWS PrivateLink: Enable private connectivity between services without exposing data to the internet.

3. Pillar: Data Protection and Encryption

Alignment with AWS Components:
• AWS Key Management Service (KMS): Manage encryption keys for AWS resources and applications.

• AWS Secrets Manager: Store and rotate secrets and credentials securely.
• AWS Certificate Manager (ACM): Manage TLS certificates for encryption in transit.
• Amazon S3 Server-Side Encryption (SSE) and RDS Encryption: Protect data at rest automatically.
• Enforce encryption policies via AWS Config rules.

4. Pillar: Least Privilege Access

Alignment with AWS Components:
• IAM Roles and Policies: Implement least privilege using fine-grained permissions.
• AWS Organizations and Service Control Policies (SCPs): Restrict access at the account and OU level.
• Attribute-Based Access Control (ABAC): Enforce dynamic permissions based on resource and user tags.
• Regularly audit IAM roles and privileges using AWS Access Analyzer.

5. Pillar: Continuous Monitoring and Analytics

Alignment with AWS Components:
• AWS CloudTrail: Capture and review all API and console activity.
• Amazon GuardDuty: Continuously monitor for threats and anomalous behavior.
• AWS Security Hub: Aggregate findings from multiple AWS security services into a single dashboard.
• AWS Detective: Analyze root causes of security events.
• Amazon CloudWatch and AWS Config: Monitor logs, metrics, and compliance drift.

6. Pillar: Automation and Orchestration

Alignment with AWS Components:
• AWS Config and AWS Organizations: Automatically enforce compliance standards and guardrails.
• AWS Lambda and EventBridge: Automate detection and remediation of security incidents.
• AWS Systems Manager (SSM): Orchestrate operational tasks and patching workflows.
• Integration with DevSecOps: Use AWS CodePipeline, CodeBuild, and CodeCommit to embed security into CI/CD processes.

Conclusion

Each AWS service aligns with specific pillars of the Zero-Trust model to build a holistic security framework.

Together, they support continuous verification, segmentation, encryption, least privilege, monitoring, and automation,
strengthening an organization's Zero-Trust posture within the AWS environment.

Testing Zero-Trust Principles on AWS

Below are step-by-step instructions to validate the Zero-Trust setup within an AWS environment.
These steps simulate real-world scenarios to test authentication, segmentation, access control, and monitoring.

Step 1: Deploy Sample Applications Across Segmented Subnets

1. Create a Virtual Private Cloud (VPC):
   - Name: MyVPC
   - CIDR Block: 10.0.0.0/16

2. Create Subnets within the VPC:
   - FrontendSubnet (10.0.1.0/24): Hosts the web application (public).
   - BackendSubnet (10.0.2.0/24): Hosts the backend service (private).

3. Launch EC2 Instances:
   - Use t3.micro instances for both subnets.
   - Assign Security Groups with appropriate inbound/outbound rules (e.g., allow HTTPS only to FrontendSubnet).
   - Deploy a simple web app on Frontend EC2 and an API service on Backend EC2.

Step 2: Attempt Unauthorized Access

1. Attempt to SSH or RDP into backend instances from an unauthorized IP address.
   - Verify access is denied by Security Group or Network Firewall rules.

2. Use IAM roles with restricted permissions and test from unauthorized accounts.
   - Confirm IAM policies enforce least privilege.

3. Validate that only approved devices and users (via Verified Access) can reach protected resources.

Step 3: Monitor AWS Security Services

1. View CloudTrail Logs:
   • Navigate to the AWS CloudTrail console and review recent activity logs for EC2 and IAM operations.

2. Check AWS GuardDuty Findings:
   • Open GuardDuty dashboard and review any findings related to unusual access or reconnaissance behavior.

3. Use AWS Security Hub and AWS Detective:
   • Review Security Hub findings and correlate them with detailed investigation in Detective.

4. Review AWS Config Compliance:
   • Confirm that encryption, IAM, and network configurations remain compliant.

Step 4: Investigate and Remediate

1. Respond to GuardDuty or Security Hub alerts using AWS Systems Manager Automation runbooks.
2. Trigger Lambda functions via EventBridge to isolate or terminate compromised instances automatically.
3. Document and analyze root causes using AWS Detective and CloudWatch logs.

Final Conclusion

By performing these tests, organizations can validate the implementation of Zero-Trust security principles in AWS.
Testing across IAM, network segmentation, and monitoring layers ensures the continuous enforcement of Zero-Trust policies,
helping maintain a robust and adaptive security posture against evolving threats.