HopkinsConnect: Cloud Application Deployment and Security Analysis

Presentation Outline (20 Slides)

Team: Mahendra and Abdulaziz

Assignment: Homework 6 - Cloud Application Deployment and Security Analysis

Slide 1: Title Slide

Content:

- Title: "HopkinsConnect: Secure Cloud Application Deployment"
- Subtitle: "OpenStack + Kubernetes Security Analysis"
- Team members: Mahendra and Abdulaziz
- Course: Cloud Security
- Date

Screenshots: None (title slide)

Slide 2: Agenda

Content:

- Application Overview & Use Case
- Deployment Architecture & Infrastructure
- Security Configuration & Best Practices
- Security Analysis & Findings
- Key Takeaways & Future Work

Screenshots: None

PART 1: APPLICATION OVERVIEW AND DEPLOYMENT

Slide 3: Application Use Case & Problem Statement

Content:

- **HopkinsConnect**: Web-based collaboration platform for JHU students
- **Key Features:**
  - User Profile Management & Discovery
  - Global Message Board (5 categories: Research, Startups, Projects, Study Groups, Other)
- **Problem:** Students work in isolated silos despite shared interests
- **Solution:** Centralized platform for collaboration discovery

Screenshots:

- **Figure 1**: Application screenshot showing message board with posts
- **Figure 2**: Application screenshot showing search functionality

Slide 4: Three-Tier Infrastructure Stack

Content:

- **Layer 1:** CloudLab Physical Infrastructure (d710 controller)
- **Layer 2:** OpenStack Cloud Platform (Zed/DevStack)
  - Services: Nova, Neutron, Magnum, Heat, Glance, Keystone
- **Layer 3:** Kubernetes Container Orchestration (v1.23.3)
  - Single-node cluster (master-only), Fedora CoreOS, Flannel CNI

Screenshots:

- **Figure 3**: OpenStack cluster showing `hopkinsconnect-k8s` in CREATE_COMPLETE/HEALTHY status
- **Figure 4**: Detailed cluster information (master/node config, Fedora CoreOS, floating IP)
- **Figure 6**: Kubernetes nodes showing master node in Ready state

Slide 5: Deployment Tools & Process

Content:

- **Containerization:** Docker 27.4.1 (multi-arch builds) → Docker Hub registry
- **Orchestration:** kubectl + Kubernetes v1.23.3
- **Deployment Phases:**
    i. Infrastructure Provisioning (CloudLab + OpenStack)
    ii. Kubernetes Cluster Creation (Magnum)
    iii. Application Containerization (Docker builds)
    iv. Kubernetes Deployment (Pods, Services, Policies)

Screenshots:

- **Figure 7**: OpenStack keypair list confirming 'mykey' registration
- **Figure 5**: OpenStack VMs showing Kubernetes master node VM (m1.medium flavor)

Slide 6: Application Deployment Status

Content:

- **Backend Pod:** 10.100.0.24:3000 (Running, non-root UID 1000)
- **Frontend Pod:** 10.100.0.18:80 (Running)
- **Storage:** PersistentVolume bound (5Gi, ReadWriteOnce)
- **Services:** Backend (ClusterIP), Frontend (LoadBalancer)
- **Network:** Flannel CNI (10.100.0.0/16 pod network)

Screenshots:

- **Figure 8**: Persistent volumes showing backend-pv bound
- **Figure 9**: `kubectl get pods -o wide` showing both pods running with IPs and resource limits

Slide 7: Architecture & Network Diagrams

Content:

- **Architecture:** CloudLab → OpenStack → Kubernetes → Application Pods
- **Network Topology:**
  - OpenStack private network (10.0.0.0/24)
  - Kubernetes Pod network (10.100.0.0/16)
  - Network policy restrictions (frontend → backend only)

Screenshots:

- **Figure 10**: Architecture diagram (from Question 3)
- **Figure 11**: Network diagram (from Question 3)

PART 2: SECURITY CONFIGURATION AND BEST PRACTICES

Slide 8: Security Measures Overview

Content:

- **11 Key Security Controls** implemented (defense-in-depth):
  i. Kubernetes Secrets Management (JWT tokens)
  ii. SSH Key-Based Authentication (4096-bit RSA)
  iii. Container Security Context (Non-root, no privilege escalation)
  iv. Resource Limits & Quotas (DoS prevention)
  v. Network Policies (Microsegmentation)
  vi. Container Image Security (Version-pinned)
  vii. Persistent Volume Security (ReadWriteOnce)
  viii. Kubernetes RBAC (Least privilege)
  ix. OpenStack Infrastructure Security
  x. Secure Container Runtime (Fedora CoreOS)
  xi. Web Application Security (XSS, SQL Injection, Headers)

Screenshots: None (overview slide)

Slide 9: Authentication & Secrets Management

Content:

- **Kubernetes Secrets:** JWT tokens stored securely (base64-encoded, RBAC-controlled)
- **SSH Key-Based Auth:** 4096-bit RSA keypair (no password authentication)
- **OpenStack Keypairs:** Infrastructure access via key-based auth

Screenshots:

- **Figure 12**:
  - `kubectl get secrets` showing app-secrets
  - `openstack keypair list` showing mykey (RSA 4096)

Slide 10: Container Security & Resource Limits

Content:

- **Security Context:**
  - Non-root execution (UID 1000)
  - `allowPrivilegeEscalation: false`
  - `capabilities.drop: ["ALL"]`
- **Resource Limits:**
  - Backend: 300m CPU / 384Mi memory
  - Frontend: 200m CPU / 256Mi memory
- **Prevents:** Privilege escalation, resource exhaustion, DoS attacks

Screenshots:

- **Figure 13**: `kubectl describe pod` showing security context
- **Figure 9 (extended)**: Deployment showing resource limits and security contexts
- Code snippet: Security context YAML from backend-deployment.yaml

Slide 11: Network Policies & Web Application Security

Content:

- **Network Policies:** Only frontend → backend communication allowed (port 3000)
- **Web Security:**
    - XSS Prevention: `escapeHtml()` function
    - SQL Injection Prevention: Parameterized queries
    - Security Headers: X-Frame-Options, X-Content-Type-Options, X-XSS-Protection
    - Helmet.js: Multiple security headers (CSP, HSTS)
    - Password Security: bcrypt hashing (10 rounds)
    - JWT Authentication: 24-hour expiration tokens

Screenshots:

- **Figure 14**: `kubectl get networkpolicy` showing hopkinsconnect-network-policy
- Code snippets: Network policy YAML, XSS prevention (app.js), SQL injection prevention (server.js)

Slide 12: Avoiding Cloud Misconfigurations

Content:

- **10 Key Practices:**
    i. Network Segmentation (Network Policies) ✓
    ii. No Default Credentials (SSH keys, Secrets) ✓
    iii. Least Privilege (Non-root, RBAC) ✓
    iv. No Public Storage (ReadWriteOnce PVs) ✓
    v. Version-Controlled Images (v1 tags) ✓
    vi. Container Security Contexts ✓
    vii. No Exposed Dashboards ✓
    viii. Immutable Infrastructure (IaC) ✓
    ix. Resource Governance (Limits) ✓
    x. Secrets Management ✓

Screenshots:

- **Figure 15**: PersistentVolume showing ReadWriteOnce access mode

- **Figure 17**: Combined validation showing resource limits, RBAC audit, and network policy test

PART 3: SECURITY ANALYSIS

Slide 13: Security Analysis Methodology & Tools

Content:

- **Three Analysis Types:**
    i. **Static Analysis:** kubesec (K8s manifests), Trivy (container images)
    ii. **Dynamic Analysis:** kubectl (runtime audit), NMAP (port scanning), Nikto (web scanning)
    iii. **Penetration Testing:** kube-hunter (K8s-specific), manual network policy testing

Screenshots:

- kubesec report output (security scores)
- Trivy scan results summary (vulnerability counts)
- kube-hunter report summary

Slide 14: Static Analysis Findings

Content:

- **kubesec Results:**
    - Security scores for manifests
    - Identified missing controls (remediated)
- **Trivy Results:**
    - **Backend:** 8 vulnerabilities (1 HIGH, 4 MEDIUM, 3 LOW)
    - **Frontend:** 21 vulnerabilities (3 CRITICAL, 2 HIGH, 9 MEDIUM, 7 LOW)
    - **Critical CVEs:** CVE-2025-49794 (libxml2), CVE-2025-58050 (pcre2)
    - **Node.js dependencies:** Clean (no vulnerabilities)

Screenshots:

- Trivy scan report snippets showing critical vulnerabilities
- kubesec security scores table

Slide 15: Dynamic Analysis Findings

Content:

- **Network Security:** 10/10 (complete isolation verified)
  - NMAP scans blocked by network policies
  - Nikto unable to reach services (positive indicator)
- **Configuration Security:** 8/10
  - Strong RBAC, network policies, resource limits
  - Capabilities dropped (post-remediation)

Screenshots:

- **Figure 17**: Combined validation showing:
  - Resource limits enforced
  - RBAC least privilege confirmed
  - Network policy blocking unauthorized access
- NMAP/Nikto output showing blocked connections

Slide 16: Penetration Testing Findings

Content:

- **kube-hunter Results:** 4 findings
  - **CAP_NET_RAW enabled** (MEDIUM) - **REMEDIATED** post-testing
  - Service account tokens accessible (MEDIUM - mitigated by RBAC)
  - AWS metadata exposure (FALSE POSITIVE - not on AWS)
  - No remote clusters discovered (GOOD - proper isolation)
- **Network Policy Validation:** Unauthorized pods blocked (DNS resolution failed)

Screenshots:

- kube-hunter report showing findings

- Before/after: Security context showing `capabilities.drop: ["ALL"]`
- Network policy test: `kubectl run test-pod` showing blocked access

Slide 17: Vulnerability Summary & Security Posture

Content:

- **Overall Security Posture:** 8.5/10
  - Good for development, requires image patching for production
- **Breakdown:**
  - Configuration Security: 8/10
  - Image Vulnerabilities: 29 total (3 critical)
  - Network Security: 10/10
  - Penetration Testing: 4 findings (1 remediated)
- **Critical Vulnerabilities:**
  - CVE-2025-49794 (libxml2 heap use-after-free)
  - CVE-2025-58050 (pcre2 heap buffer overflow)
  - **Remediation:** Update Alpine base images to latest patched versions

Screenshots:

- Summary table/chart of findings by severity
- Trivy vulnerability breakdown chart

Slide 18: Security Strengths & Areas for Improvement

Content:

- **Strengths:**
  - Strong network segmentation (10/10)
  - Effective RBAC policies
  - Resource limits enforced
  - Non-root containers
  - Secrets management
  - Web application security (XSS, SQL injection prevention)

- **Areas for Improvement:**
  - Update base images (critical CVEs)
  - Consider read-only root filesystems where possible
  - Regular vulnerability scanning in CI/CD

Screenshots: None (summary slide)

CONCLUSION

Slide 19: Key Takeaways

Content:

- Successfully deployed secure cloud application on OpenStack + Kubernetes
- Implemented 11 security controls following defense-in-depth principles
- Comprehensive security analysis using static, dynamic, and penetration testing
- Identified and documented vulnerabilities with remediation plans
- Demonstrated real-world cloud security best practices
- **Deployment Time:** ~90 minutes (including troubleshooting)

Screenshots:

- **Figure 10**: Architecture diagram (reuse from earlier)
- Final deployment status showing all components operational

Slide 20: Future Work & Q&A

Content:

- **Immediate:** Patch base images (critical CVEs)
- **Short-term:** Multi-node cluster, automated vulnerability scanning in CI/CD
- **Long-term:** Service mesh (Istio/Linkerd), Pod Security Standards, compliance automation
- **Lessons Learned:**

    - Infrastructure constraints (single-node cluster)
    - Security trade-offs (writable root filesystem)
    - Network policies require careful testing
    - Regular vulnerability scanning essential

Screenshots: None

Content:

- Questions?
- Contact information
- References: OpenStack, Kubernetes, CloudLab Documentation

Presentation Tips

Visual Flow:

- **Part 1 (Slides 3-7):** Application and deployment (~7 minutes)
- **Part 2 (Slides 8-12):** Security configuration (~5 minutes)
- **Part 3 (Slides 13-18):** Security analysis (~6 minutes)
- **Conclusion (Slides 19-20):** Summary & Q&A (~2 minutes)
- **Total:** ~20 minutes

Important Notes:

- **Include JHEDIDs** in all screenshots (as per assignment requirements)
- **Use original screenshots** (not edited with added text)
- **Reference figure numbers** consistently (Figure 1-17 from homework)
- **Keep slides concise** - use bullet points, not paragraphs
- **Highlight key findings** with visual emphasis (colors, bold text)
- **Group related screenshots** on the same slide when possible
- **Use Figure 17** strategically - it's a powerful combined validation screenshot