HopkinsConnect Presentation Script

Team: Mahendra and Abdulaziz

Presentation Time: ~20 minutes
Format: Alternating slides between presenters

Slide 1: Title Slide

Presenter: Mahendra

Script:
"Good [morning/afternoon], everyone. I'm Mahendra, and this is my teammate Abdulaziz. Today we'll be presenting our work on HopkinsConnect, a secure cloud application deployment using OpenStack and Kubernetes. We'll walk you through the application, our deployment architecture, security configurations, and comprehensive security analysis."

[Pause for slide to be visible]

Slide 2: Agenda

Presenter: Abdulaziz

Script:
"Let me outline what we'll cover today. First, we'll give you an overview of our application and its use case. Then, Mahendra will walk you through our deployment architecture and infrastructure setup. Next, we'll discuss the security measures we implemented, followed by our security analysis findings. Finally, we'll share key takeaways and future work."

[Pause briefly]

"Let's begin with the application overview."

Slide 3: Application Use Case & Problem Statement

Presenter: Mahendra

Script:
"HopkinsConnect is a web-based collaboration platform designed specifically for Johns Hopkins University students. As you can see from the screenshots, it provides three main features: user profile management and discovery, a global message board with five categories - Research Collaboration, Startup Co-founders, Class Projects, Study Groups, and Other - and profile search functionality."

"The problem we're solving is that JHU students often work in isolated silos despite having shared interests and complementary skills. Traditional methods like email lists or random encounters are inefficient. Our solution is a centralized platform that makes it easy for students to discover collaboration opportunities."

[Point to screenshots]

"Here you can see the message board in action, and the search functionality that helps students find relevant posts and profiles."

Slide 4: Three-Tier Infrastructure Stack

Presenter: Abdulaziz

Script:
"Now let me explain our infrastructure architecture. We deployed HopkinsConnect using a three-tier stack."

"At the bottom layer, we have CloudLab's physical infrastructure - specifically a d710 controller node that provides bare-metal compute resources."

"The middle layer is OpenStack, running the Zed release via DevStack. We leveraged several key services: Nova for compute, Neutron for networking, Magnum for container orchestration, Heat for orchestration, Glance for image management, and Keystone for identity services."

"At the top layer, we have Kubernetes version 1.23.3 running as a single-node cluster on Fedora CoreOS, using Flannel as our CNI for pod networking."

[Point to screenshots]

"As you can see from these screenshots, our OpenStack cluster shows the hopkinsconnect-k8s cluster in CREATE_COMPLETE and HEALTHY status, and the Kubernetes master node is ready and operational."

Slide 5: Deployment Tools & Process

Presenter: Mahendra

Script:
"For containerization, we used Docker version 27.4.1 with multi-architecture builds, pushing our images to Docker Hub. Our orchestration was handled through kubectl and Kubernetes v1.23.3."

"Our deployment followed four main phases. First, we provisioned the infrastructure on CloudLab and set up OpenStack. Second, we created the Kubernetes cluster using Magnum. Third, we containerized our application with Docker builds. And finally, we deployed everything to Kubernetes - pods, services, and network policies."

[Point to screenshots]

"Here you can see the OpenStack keypair we created for authentication, and the VM that hosts our Kubernetes master node."

Slide 6: Application Deployment Status

Presenter: Abdulaziz

Script:
"Let me show you the current state of our deployment. Our backend pod is running at IP 10.100.0.24 on port 3000, configured to run as a non-root user with UID 1000. The frontend pod is at 10.100.0.18 on port 80."

"We've set up persistent storage with a 5-gigabyte volume using ReadWriteOnce access mode, which is currently bound to our backend pod. For services, we have a ClusterIP service for the backend and a LoadBalancer service for the frontend."

"The entire setup uses Flannel CNI with a pod network range of 10.100.0.0/16."

[Point to screenshots]

"These screenshots confirm that our persistent volume is bound and both pods are running with their assigned IPs and resource limits."

Slide 7: Architecture & Network Diagrams

Presenter: Mahendra

Script:
"Here's a visual representation of our complete architecture. As you can see, the flow goes from CloudLab's physical infrastructure, through OpenStack's virtualization layer, into our Kubernetes cluster, and finally to our application pods."

"The network diagram shows our network topology. We have an OpenStack private network at 10.0.0.0/24, a Kubernetes pod network at 10.100.0.0/16, and network policy restrictions that ensure only the frontend can communicate with the backend."

"This segmentation is a key security feature that we'll discuss more in the security section."

Slide 8: Security Measures Overview

Presenter: Abdulaziz

Script:
"Now let's move to security. We implemented 11 key security controls following defense-in-depth principles."

"These include Kubernetes secrets management for JWT tokens, SSH key-based authentication using 4096-bit RSA keys, container security contexts with non-root execution and no privilege escalation, resource limits to prevent DoS attacks, network policies for microsegmentation, version-pinned container images, secure persistent volumes, Kubernetes RBAC for least privilege, OpenStack infrastructure security, a secure container runtime with Fedora CoreOS, and comprehensive web application security measures including XSS and SQL injection prevention."

"We'll dive deeper into specific controls in the next few slides."

Slide 9: Authentication & Secrets Management

Presenter: Mahendra

Script:
"Let's start with authentication and secrets management. For Kubernetes, we store JWT tokens securely using Kubernetes Secrets, which are base64-encoded and access-controlled through RBAC."

"For infrastructure access, we use SSH key-based authentication with a 4096-bit RSA keypair, completely eliminating password-based authentication. Similarly, OpenStack keypairs are used for all infrastructure access."

[Point to screenshots]

"As shown in these screenshots, we have our app-secrets stored in Kubernetes, and our OpenStack keypair registered. This approach prevents credential exposure and brute-force attacks."

Slide 10: Container Security & Resource Limits

Presenter: Abdulaziz

Script:
"For container security, we configured our backend to run as a non-root user with UID 1000. We've disabled privilege escalation and dropped all Linux capabilities. This follows the principle of least privilege."

"We've also set strict resource limits: the backend is limited to 300 millicores of CPU and 384 megabytes of memory, while the frontend is limited to 200 millicores and 256 megabytes."

"These measures prevent privilege escalation attacks, resource exhaustion, and denial of service attacks."

[Point to screenshots]

"The security context configuration and resource limits are visible in these deployment descriptions."

Slide 11: Network Policies & Web Application Security

Presenter: Mahendra

Script:
"Our network policies enforce strict microsegmentation. The backend on port 3000 can only receive traffic from the frontend - no other pods can access it directly. The frontend on port 80 can receive traffic from anywhere, as expected for a web application."

"For web application security, we implemented multiple layers. We use an escapeHtml function to prevent XSS attacks, parameterized queries to prevent SQL injection, security headers like X-Frame-Options and X-Content-Type-Options, Helmet.js middleware for additional headers, bcrypt password hashing with 10 rounds, and JWT authentication with 24-hour expiration tokens."

[Point to code snippets]

"Here you can see examples of our network policy configuration and the security code implementations."

Slide 12: Avoiding Cloud Misconfigurations

Presenter: Abdulaziz

Script:
"We followed 10 key practices to avoid common cloud misconfigurations. These include network segmentation through network policies, no default credentials using SSH keys and secrets, least privilege with non-root containers and RBAC, no public storage with ReadWriteOnce persistent volumes, version-controlled images using specific tags instead of 'latest', proper container security contexts, no exposed dashboards, immutable infrastructure through Infrastructure as Code, resource governance with limits, and proper secrets management."

[Point to screenshots]

"These screenshots demonstrate our persistent volume configuration and combined validation showing resource limits, RBAC audit, and network policy enforcement."

Slide 13: Security Analysis Methodology & Tools

Presenter: Mahendra

Script:
"Now let's move to our security analysis. We used three types of analysis."

"For static analysis, we used kubesec to scan our Kubernetes manifests and Trivy to scan our container images. For dynamic analysis, we used kubectl for runtime configuration audits, NMAP for port scanning, and Nikto for web vulnerability scanning. For penetration testing, we used kube-hunter, which is specifically designed for Kubernetes clusters, and manual network policy testing."

"This comprehensive approach allowed us to identify vulnerabilities at different stages - before deployment, during runtime, and through active exploitation attempts."

Slide 14: Static Analysis Findings

Presenter: Abdulaziz

Script:
"Let me share our static analysis results. Kubesec gave our backend deployment a security score of 5 out of 10 points, identifying several security controls that we then remediated."

"Trivy scanning revealed 8 vulnerabilities in our backend image - 1 HIGH, 4 MEDIUM, and 3 LOW severity. More concerning, our frontend image had 21 vulnerabilities, including 3 CRITICAL, 2 HIGH, 9 MEDIUM, 7 LOW, and 2 UNKNOWN."

"The critical vulnerabilities are CVE-2025-49794 in libxml2, which is a heap use-after-free vulnerability, and CVE-2025-58050 in pcre2, which is a heap buffer overflow. Importantly, all our Node.js application dependencies came back clean with no vulnerabilities."

[Point to screenshots]

"These screenshots show the kubesec security score and the Trivy vulnerability breakdown, highlighting the critical CVEs that require immediate attention."

Slide 15: Dynamic Analysis Findings

Presenter: Mahendra

Script:
"Our dynamic analysis showed excellent results for network security - we scored 10 out of 10. NMAP port scans were completely blocked by our network policies, and Nikto was unable to reach our services. These 'failures' are actually positive security indicators, demonstrating that our network policies are working correctly."

"For configuration security, we scored 8 out of 10. We have strong RBAC policies, effective network policies, proper resource limits, and we've dropped all capabilities after remediation."

[Point to screenshots]

"This combined validation screenshot shows resource limits being enforced, RBAC least privilege confirmed, and network policies successfully blocking unauthorized access."

Slide 16: Penetration Testing Findings

Presenter: Abdulaziz

Script:
"Kube-hunter identified 4 findings during penetration testing. The first was CAP_NET_RAW being enabled, which is a MEDIUM severity finding. We remediated this post-testing by adding capabilities.drop with ALL to our security context."

[Point to kube-hunter report screenshot]

"This kube-hunter report shows all four findings. As you can see, the first finding is CAP_NET_RAW enabled, which could allow network attacks if a pod is compromised."

[Point to security context screenshot if included]

"Here's the remediation we implemented. This security context shows that we've added capabilities.drop with ALL, which removes the CAP_NET_RAW capability and addresses this vulnerability. This was added post-kube-hunter testing based on the finding."

"Second, service account tokens are accessible, which is also MEDIUM severity, but this is significantly mitigated by our restrictive RBAC policies that limit what the tokens can actually do."

"Third, there was an AWS metadata exposure finding, but this is a false positive since we're not running on AWS - we're on CloudLab infrastructure."

"Finally, kube-hunter confirmed that no remote clusters were discovered, which is good - it shows proper isolation."

[If network policy screenshot is shown]

"Here's our network policy configuration. As you can see, we have two key ingress rules. First, the frontend on port 80 can receive traffic - this allows external users to access our web application through the LoadBalancer service. Second, and more importantly for security, the backend on port 3000 can only receive traffic from pods labeled as frontend. This means no other pods in the cluster can directly access the backend."

[If network policy test is shown]

"We validated this by attempting unauthorized pod-to-pod communication. As expected, the connection was blocked, confirming that our network segmentation is working correctly. The network policies ensure that only the frontend can communicate with the backend, preventing lateral movement in case of a compromise."

"Additionally, we allow all egress traffic, though this could be restricted further for even tighter security."

Slide 17: Vulnerability Summary & Security Posture

Presenter: Mahendra

Script:
"Overall, we rate our security posture at 8.5 out of 10. This is good for a development environment, but would require image patching before production deployment."

"Breaking it down: Configuration security is 8 out of 10, we have 29 total image vulnerabilities with 3 critical ones, network security is perfect at 10 out of 10, and penetration testing found 4 issues with 1 already remediated."

"The critical vulnerabilities - CVE-2025-49794 in libxml2 and CVE-2025-58050 in pcre2 - require updating our Alpine base images from versions 3.21.3 and 3.22.0 to the latest patched versions."

[Point to screenshots]

"This summary table shows the breakdown of vulnerabilities by severity across frontend, backend, and penetration testing findings."

Slide 18: Security Strengths & Areas for Improvement

Presenter: Abdulaziz

Script:
"Let me summarize our key strengths and areas for improvement."

"Our strengths include strong network segmentation with a perfect 10 out of 10 score, effective RBAC policies, enforced resource limits, non-root containers, proper secrets management, and comprehensive web application security with XSS and SQL injection prevention."

"Areas for improvement include updating our base images to patch the critical CVEs, considering read-only root filesystems where possible, and implementing regular vulnerability scanning in a CI/CD pipeline."

"These improvements would bring our security posture from good for development to production-ready."

Slide 19: Key Takeaways

Presenter: Mahendra

Script:
"To summarize our work: We successfully deployed a secure cloud application on OpenStack and Kubernetes. We implemented 11 security controls following defense-in-depth principles. We

conducted comprehensive security analysis using static, dynamic, and penetration testing methods. We identified and documented all vulnerabilities with clear remediation plans. And we demonstrated real-world cloud security best practices."

"Our total deployment time was approximately 90 minutes, including troubleshooting."

[Point to architecture diagram]

"This architecture diagram represents the complete stack we built and secured."

Slide 20: Future Work & Q&A

Presenter: Abdulaziz

Script:
"Looking ahead, our immediate priority is patching the base images to address the critical CVEs. In the short term, we'd like to deploy a multi-node cluster for high availability and implement automated vulnerability scanning in CI/CD. Long-term goals include implementing a service mesh like Istio or Linkerd for advanced network policies, adopting Pod Security Standards, and automating compliance checks."

"Key lessons we learned include working within infrastructure constraints - we had to use a single-node cluster due to CloudLab limitations. We also learned about security trade-offs, like needing a writable root filesystem for application functionality. Network policies require careful testing, and regular vulnerability scanning is absolutely essential."

"Thank you for your attention. We're now happy to take any questions."

[Pause for questions]

Presentation Tips

Timing Guidelines:

- **Part 1 (Slides 3-7):** ~7 minutes (Application and deployment)
- **Part 2 (Slides 8-12):** ~5 minutes (Security configuration)

- **Part 3 (Slides 13-18):** ~6 minutes (Security analysis)
- **Conclusion (Slides 19-20):** ~2 minutes (Summary & Q&A)
- **Total:** ~20 minutes

Transition Phrases:

- "Now let me hand it over to [name]..."
- "Let me show you..."
- "As you can see from this screenshot..."
- "Moving on to..."
- "Let's now discuss..."

Speaking Tips:

- **Point to screenshots** when referencing them
- **Pause briefly** after showing new slides
- **Speak clearly** and at a moderate pace
- **Make eye contact** with the audience
- **Use hand gestures** to emphasize key points
- **Practice transitions** between slides

Backup Notes:

- If you forget a detail, refer to the slide content
- If asked a question you don't know, say "That's a great question. Let me think about that..." or "We can discuss that after the presentation"
- If technology fails, continue with the content - the slides are just visual aids

Slide Assignment Summary

Mahendra presents: Slides 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 (10 slides)
Abdulaziz presents: Slides 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 (10 slides)

Total: 20 slides, evenly distributed