Presentation Script: Vendor Risk Digital Twin

Total Time: 10 minutes

Slide 1: Title (15 seconds)

Speaker: Cliff

[Cliff]
"Good morning/afternoon. I'm Clifford Odhiambo, and I'm here with Mahendra Shahi and Jalil Rezek. Today we'll present our work on the Vendor Risk Digital Twin: A Cloud-Native Framework for Predicting Third-Party Failure Impact."

Slide 2: Problem Statement & Motivation (1.5 minutes)

Speaker: Cliff

[Cliff]
"Let me start with the problem we're addressing. Modern cloud-native organizations depend on 30 to 50 third-party SaaS vendors. When these vendors fail, the impact is severe - we're talking an average cost of 500 thousand to 2 million dollars per major vendor failure.

Current vendor risk management tools are reactive. They rely on static questionnaires and point-in-time assessments. They can't map vendor dependencies to actual cloud resources or business processes.

This matters because regulatory requirements like DORA and NIS2 now mandate resilience testing. Organizations need predictive capabilities, not reactive checklists.

As one industry expert, John Zehnpfennig from CGI Federal, noted: 'Especially on the contract, the legality side, the trade compliance, the contracts, procurements, that's a lot of manual work and sometimes they get tricked.'"

Slide 3: Background / Prior Work (1 minute)

Speaker: Cliff

[Cliff]
"Let's look at what exists today. GRC platforms require manual data entry and offer no simulation capabilities. Security ratings provide external scores but can't map to infrastructure. Risk quantification tools focus only on financial impact, missing operational and compliance dimensions.

The gap is clear: no existing solution combines automated cloud-native discovery, graph-based dependency modeling, real-time failure simulation, and multi-dimensional impact prediction.

As Anurag Baral from Carahsoft observed: 'Most companies are still struggling to implement basic automation... so a dynamic, predictive approach is exactly where the market is heading, even if it still feels futuristic to some.'

Our work aligns with the emerging GRC 7.0 paradigm, which emphasizes digital twins and forward-looking risk intelligence, as described by Michael Rasmussen."

Slide 4: Design / Approach (1 minute)

Speaker: Mahendra

[Mahendra]
"Thank you, Cliff. Our solution is built on a cloud-native, event-driven architecture with a 4-layer design. We leverage multiple GCP services and follow four key design principles: serverless backend with auto-scaling, event-driven processing with Pub/Sub, graph-based modeling with Neo4j, and automated discovery through GCP API integration.

Our graph model uses four node types - Vendor, Service, BusinessProcess, and ComplianceControl - connected through DEPENDS_ON, SUPPORTS, and SATISFIES relationships. Let me show you the detailed architecture now."

Slide 5: Architecture Diagrams (2 minutes)

Speaker: Mahendra

[Mahendra]
"Now let's dive into the architecture details. Figure 1 shows our 4-layer architecture diagram.

Starting at the top, the Presentation Layer includes our Web Dashboard built with Node.js, the Neo4j Browser for graph visualization, and REST API endpoints.

The Application Layer contains our core business logic: the Discovery Function using Cloud Functions Gen2 that queries GCP APIs, the Simulation Service running on Cloud Run for impact calculations, the Graph Loader that automatically updates Neo4j, and our CI/CD Pipeline using Cloud Build.

The Data Layer stores everything: Neo4j for our graph database, Cloud Storage for discovery results, and BigQuery for analytics and historical tracking.

Finally, the External Systems layer includes GCP APIs for Functions and Cloud Run services, as well as compliance frameworks like SOC 2, NIST, and ISO.

You can see the data flow: the Dashboard connects to the REST API, which triggers the Simulation Service and Discovery Function. Discovery queries GCP APIs, stores results in Cloud Storage, and triggers the Graph Loader via Pub/Sub. The Graph Loader updates Neo4j, and the Simulation Service queries Neo4j and writes analytics to BigQuery.

Figure 2 shows our graph data model in detail. We have four node types: Vendors like Stripe and Auth0, Services like payment-api, Business Processes like checkout, and Compliance Controls. The relationships show how services depend on vendors, how services support business processes, and how vendors satisfy compliance controls. For example, payment-api depends on Stripe, which supports the checkout process, and Stripe satisfies SOC 2 compliance controls.

This graph structure enables us to traverse dependencies and calculate cascading impacts when a vendor fails."

Slide 6: Implementation (1.5 minutes)

Speaker: Mahendra

[Mahendra]
"For implementation, we built a web dashboard using Node.js and Express with a RESTful API architecture. The frontend is HTML, CSS, and JavaScript with a responsive UI.

The backend integrates directly with Neo4j for real-time graph queries. We expose REST API endpoints for vendor simulation and graph statistics. Key features include vendor selection, failure duration configuration, multi-dimensional impact visualization, and actionable recommendations.

For data sources, operational and financial metrics are configurable via config.yaml. Compliance data comes from our compliance_frameworks.yaml file, which maps vendors to SOC 2, NIST, and ISO 27001 controls. For example, Stripe maps to SOC 2 control CC6.6 for transmission security with a weight of 0.12."

Slide 7: Results (1 minute)

Speaker: Jalil

[Jalil]
"Thank you, Mahendra. Let me present our functional evaluation results. Our vendor detection uses pattern-based matching for known vendors like Stripe, Auth0, and SendGrid. We've verified our simulation accuracy through comprehensive unit tests covering all calculation logic.

We have unit and integration tests for discovery, simulation, and impact calculations. For scaling, we've configured auto-scaling with Cloud Run supporting 0 to 10 container instances, and Cloud Functions auto-scaling based on Pub/Sub message volume."

Slide 8: Results Visualizations (1 minute)

Speaker: Jalil

[Jalil]
"Here are our key visualizations. Figure 3 shows our Neo4j graph displaying the relationships between vendors, compliance controls, and cloud services. This demonstrates how we've successfully mapped real GCP infrastructure to vendor dependencies.

Figure 4 shows a simulation result where we simulated Auth0 going down for 4 hours. The report shows the full multi-dimensional impact - operational, financial, and compliance - providing actionable insights for risk management."

Slide 10: Dashboard Performance Metrics (30 seconds)

Speaker: Jalil

[Jalil]
"Figure 5 shows our dashboard interface for tracking performance and metrics. This provides real-time visibility into vendor dependencies and risk posture."

Slide 11: Remaining Issues & Future Directions (45 seconds)

Speaker: Mahendra

[Mahendra]
"Our current limitations include pattern-based vendor detection with approximately a 5% false negative rate, and we're currently GCP-only with no AWS or Azure support yet.

For future directions, we plan to add multi-cloud support for AWS and Azure, implement ML-based vendor detection to improve accuracy, integrate with Vertex AI for machine learning capabilities, and reach out to small government federal contractors who could benefit from this solution."

Slide 12: Summary & Key Takeaways (1 minute)

Speaker: Mahendra

[Mahendra]
"To summarize: we addressed the problem that organizations cannot predict vendor failure impact, and current tools are reactive rather than predictive.

Our solution provides automated cloud-native discovery from GCP, graph-based dependency modeling, and real-time failure simulation.

As Michael Rasmussen, a GRC analyst, puts it: 'Digital twins are not dashboards. They are strategic instruments of foresight. They empower GRC to shift from accountability to adaptability, from control to intelligence.'

Our proven integration approach demonstrates automated data ingestion at scale. As shown in Figure 6, our framework can serve as an augmented intelligence layer for existing GRC platforms like Archer, MetricStream, and ServiceNow, combining automated discovery with enterprise GRC workflows."

Slide 13: Thank You (10 seconds)

Speaker: Mahendra

[Mahendra]
"Thank you for your attention. We're happy to take any questions."

Timing Summary

Slide	Speaker	Time	Cumulative
1	Cliff	15s	0:15
2	Cliff	1:30	1:45
3	Cliff	1:00	2:45
4	Mahendra	1:00	3:45
5	Mahendra	2:00	5:45
6	Mahendra	1:30	7:15
7	Jalil	1:00	7:15
8	Jalil	1:00	8:15

Slide	Speaker	Time	Cumulative
10	Jalil	0:30	8:45
11	Mahendra	0:45	9:30
12	Mahendra	1:00	10:30
13	Mahendra	0:10	10:40

Total: ~10 minutes 40 seconds (includes buffer for transitions)

Presentation Tips

1. **Practice transitions** - Smooth handoffs between speakers
2. **Eye contact** - Look at the audience, not just the slides
3. **Pace** - Speak clearly and at a moderate pace
4. **Visuals** - Point to specific figures when discussing them
5. **Q&A preparation** - Be ready to discuss technical details, limitations, and future work

Key Talking Points to Emphasize

- **Problem:** Reactive tools, no cloud integration, high cost of failures
- **Solution:** Automated discovery, graph modeling, real-time simulation
- **Innovation:** First solution combining all four capabilities
- **Impact:** Predictive risk management, regulatory compliance support
- **Integration:** Can enhance existing GRC platforms