Table 3. **Orchestrator with SQL Agent, Explicit Attacks** - average benign query accuracy (BA), robust benign query accuracy (RA), expected number of queries for a successful attack (E).

Model	Employee Toy			Employee Medium			Employee Big		
	BA	RA	E	BA	RA	E	BA	RA	E
gpt-4.1-mini	100.0%	84.0%	6	100.0%	73.6%	4	96.0%	71.6%	6
gpt-4.1	100.0%	75.8%	23	98.0%	63.4%	17	92.0%	61.6%	18
o4-mini	100.0%	90.6%	500	100.0%	84.6%	∞	100.0%	78.2%	∞
claude-sonnet-4	100.0%	93.6%	∞	100.0%	93.6%	∞	100.0%	93.6%	∞
gemini-2.5-flash	100.0%	75.4%	17	100.0%	61.8%	17	100.0%	62.2%	9

* ∞ indicates the injections were unsuccessful

Table 4. **Orchestrator with SQL Agent, Implicit Attacks** - average benign query accuracy (BA), robust benign query accuracy (RA), expected number of queries for a successful attack (E).

Model	Employee Toy			Employee Medium			Employee Big		
	BA	RA	E	BA	RA	E	BA	RA	E
gpt-4.1-mini	100.0%	87.2%	7	100.0%	76.2%	7	96.0%	73.8%	9
gpt-4.1	100.0%	77.8%	42	98.0%	64.6%	34	92.0%	64.8%	56
o4-mini	100.0%	90.4%	∞	100.0%	84.0%	∞	100.0%	76.4%	∞
claude-sonnet-4	100.0%	95.4%	∞	100.0%	95.2%	∞	100.0%	95.0%	∞
gemini-2.5-flash	100.0%	76.4%	18	100.0%	63.6%	20	100.0%	66.4%	14

* ∞ indicates the injections were unsuccessful

Table 5. **Category of Attacks** - the average expected number of queries for a successful attack (E) across three database sizes (toy, medium, big) against explicit and implicit .

Model	Attack Category	Explicit Attacks			Implicit Attacks		
		Toy	Medium	Big	Toy	Medium	Big
gpt-4.1-mini	Blocking	5	4	5	7	6	10
	Compliance	4	3	4	4	5	5
	Fixed-Structure	8	9	15	9	17	22
	Combined	10	6	3	25	8	25
gpt-4.1	Blocking	17	9	10	50	22	∞
	Compliance	14	22	19	50	30	38
	Fixed-Structure	∞	∞	∞	150	∞	∞
	Combined	25	10	9	10	17	10
o4-mini	Blocking	150	∞	∞	∞	∞	∞
gemini-2.5-flash	Blocking	50	150	19	75	75	75
	Compliance	10	8	4	12	14	9
	Fixed-Structure	50	30	38	25	25	19
	Combined	7	10	6	8	8	8

* ∞ indicates the injections were unsuccessful. All injections for claude-sonnet-4 and all non-Blocking injections for o4-mini were unsuccessful, which are omitted to save space.

Database size does not appear to impact attack success. We observe no consistent relationship between database size and the expected number of queries required for a successful attack ($\mathbb{E}_{\text{attacks}}$). For example, `gpt-4.1-mini` has an $\mathbb{E}_{\text{attacks}}$ of 6, 4, 6 (explicit version) and 7, 7, 9 (implicit version) across the three database sizes. This suggests that even large production-scale databases, when interacting with an SQL agent, may be at risk.

Database size may influence benign query accuracy under attack (RA). Across all models and both explicit and implicit settings, the highest RA is observed with the Toy-sized database. RA tends to decline slightly as database size increases (e.g. in Table 4, `o4-mini` goes from 90.4% to 84.0% to 76.4% as database size increases), suggesting that injections interfere more with normal system behaviour when the database is bigger. However, this also implies that such disruptions, and therefore the attacks, may be more noticeable in bigger databases.

Table 6. **Mixing different models** - the expected number of queries for a successful attack (E) when we choose different models for the orchestrator and the downstream agents.

Orchestrator	Downstream Agents	Explicit			Implicit		
Model	Model	Toy	Medium	Big	Toy	Medium	Big
gpt-4.1-mini	claude-sonnet-4	84	∞	250	∞	∞	∞
claude-sonnet-4	gpt-4.1-mini	14	15	16	6	6	9

* ∞ indicates the injections were unsuccessful

Table 7. **Implicit Attacks** - Average expected number of queries for a successful attack (E). **PS:** Pure SQL Agent **OS:** Orchestrator with SQL Agent **ON:** Orchestrator with SQL and Notification Agent **OA:** Orchestrator with Additional Agents (Report Agent and Scheduling/Calendar Agent).

	Toy				Medium				Big			
	PS	OS	ON	OA	PS	OS	ON	OA	PS	OS	ON	OA
gpt-4.1-mini	12	8	7	6	25	10	7	6	42	13	13	12
gpt-4.1	167	42	42	56	250	46	34	28	167	63	56	42
o4-mini	250	225	∞	∞	250	500	∞	∞	∞	450	∞	∞
sonnet-4	∞	78	∞	∞	∞	450	∞	∞	∞	500	∞	∞
gem-2.5-flash	23	24	18	46	30	41	20	42	39	62	14	33

* ∞ indicates injections were unsuccessful

Effectiveness of attack categories varies by model. As shown in Table 5, the Compliance and Combined categories are notably more effective against `gemini-2.5-flash`, often requiring 3–15× fewer queries (E) (e.g. Explicit Blocking category has an E of 150 vs for Compliance and Combined categories have an E of 8 and 10 respectively). For `o4-mini`, only the Blocking category achieves success. Fixed-Structure appears to be the least effective category .

Downstream agents may affect overall setup vulnerability more. We conduct a focused experiment using our most robust model, `claude-sonnet-4`, and our most vulnerable model, `gpt-4.1-mini`, as shown in Table 6. When `gpt-4.1-mini` serves as the SQL agent, it paraphrases the adversarial instruction in a way that obscures its intent. As a result, the Orchestrator agent, `claude-sonnet-4`, cannot detect the threat, since it receives no direct indication of the malicious content.

5.3. Susceptibility as number of agents varies

Apart from the base setup, Orchestrator with SQL and Notification Agent (ON), we run analogous evaluations over three other setups (see Table 7), each of which has a different number of downstream agents. In addition to the SQL and Notification Agents, the Orchestrator with Additional Agents (OA) setup introduces a Report Agent and a Scheduling/Calendar Agent that assist in generating a report according to company guidelines and scheduling meetings, respectively. The Orchestrator Agent with SQL Agent (OS) gets rid of all of the downstream agents except the SQL Agent, and the Pure SQL Agent (PS) goes a step further and removes the orchestrator in between the user and the SQL Agent. Identical attacks are used in the ON and OA Settings. Since OS and PS settings lack a Notification Agent to exfiltrate data, we slightly modify the adversary's objective to copy the private data to the public database, instead of exfiltrating via the Notification Agent. Thus, the attacks for these two setups are modified to reflect this change, but the attack categories and how each attack is phrased are kept consistent across all 4 setups.

We observe that while certain models such as `gpt-4.1-mini` are consistently vulnerable across all 4 setups, most other models follow an inconsistent pattern, suggesting that safety in single-agent settings isn't indicative of multi-agent settings (and vice-versa). Another interesting finding is that `claude-sonnet-4` is more vulnerable to the attack when instructed to copy to a public database rather than sending the private information via email. This is possibly since `claude-sonnet-4` has been fine-tuned for safety against phishing attempts that contain a suspicious email, although this is only a hypothesis. In Appendix E, the Table 12 shows results analogous to Table 7, but for explicit attacks instead.

6. Conclusion

Data leakage risks in orchestrator multi-agent systems are underexplored. We present OMNI-Leak, a benchmark for measuring model susceptibility and factors influencing vulnerability, leaving adaptation to other architectures for future work. Even with strong practices like SQL access controls, systems remain exposed to indirect prompt injections. Low attack success rates can be misleading: in a 100-person company, a 1/500 rate could leak sensitive data within five days, and for weaker models or larger firms, within minutes. Safety-focused models appear less vulnerable, though mitigation strategies and comparisons are left to future work.

Impact Statement

This paper presents work aimed at demonstrating the real possibility of underexplored harms from orchestrator multi-agent systems. We recommend that engineers prepare defenses before real harm occurs. In particular, engineers working with such systems should heed the following:

- Access control is insufficient as a safeguard measure if data entry itself is not adequately sanitised and monitored
- Adding a monitor or filter at every step of the agentic system, i.e. before and after each user-agent interaction and inter-agent communication, would help flag any malicious activity
- Enabling monitors or agents to alert a human to suspicious activity immediately is another defensive measure we'd recommend

Acknowledgments

Adel Bibi and Alasdair Paren would like to acknowledge Toyota Motor Europe (TME) for funding the project.

References

Accelirate. Ai agent orchestration: Managing multi-agent systems for business efficiency. https://www.accelirate.com/ai-agent-orchestration/, 2025. Blog post, published January 24, 2025.

Adobe. Adobe launches adobe experience platform agent orchestrator for businesses. https://news.adobe.com/news/2025/03/adobe-launches-adobe-experience-platform-agent-orchestrator-for-businesses, 2025. Accessed: 2025-04-27.

AI Security Institute, U. Inspect AI: Framework for Large Language Model Evaluations, 2024. URL https://github.com/UKGovernmentBEIS/inspect_ai.

Andriushchenko, M., Souly, A., Dziemian, M., Duenas, D., Lin, M., Wang, J., Hendrycks, D., Zou, A., Kolter, J. Z., Fredrikson, M., Gal, Y., and Davies, X. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, 2025.

Anthropic. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku, 2024. URL https://www.anthropic.com/news/3-5-models-and-computer-use. Accessed: 2025-04-27.

Anwar, U., Saparov, A., Rando, J., Paleka, D., Turpin, M., Hase, P., Lubana, E. S., Jenner, E., Casper, S., Sourbut, O., Edelman, B. L., Zhang, Z., Günther, M., Korinek, A., Hernandez-Orallo, J., Hammond, L., Bigelow, E., Pan, A., Langosco, L., Korbak, T., Zhang, H., Zhong, R., hÉigeartaigh, S. O., Recchia, G., Corsi, G., Chan, A., Anderljung, M., Edwards, L., Petrov, A., de Witt, C. S., Motwan, S. R., Bengio, Y., Chen, D., Torr, P. H. S., Albanie, S., Maharaj, T., Foerster, J., Tramer, F., He, H., Kasirzadeh, A., Choi, Y., and Krueger, D. Foundational challenges in assuring alignment and safety of large language models. *Transactions on Machine Learning Research*, 2024.

AWS Labs. Agent squad. https://github.com/awslabs/agent-squad, 2025. Accessed: 2026-01-28.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Camel-AI. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. https://github.com/camel-ai/owl, 2025. Accessed: 2025-04-27.

Emergence AI. Data science – generate a customer satisfaction model – generate a predicted product defection model. https://www.youtube.com/watch?v=-F1ISl_x_Mw&list=PLWOqB67d9l-PXeyTTcBaRNNif8TP-_Afc&index=5, 2024. YouTube video.

Google. Adk-samples: Data science agent. https://github.com/google/adk-samples/tree/9c1c5b78212a73c57643703b9a60ff658202de80/python/agents/data-science, 2025a. GitHub directory at specific commit, example agent implementation.

Google. Agent development kit documentation. https://google.github.io/adk-docs/, 2025b. Online documentation for Google's Agent Development Kit.

Google Cloud. Vertex ai agent builder. https://cloud.google.com/products/agent-builder?hl=en, 2025. Product information page (last updated August 14, 2025).