ChatGPT

Linking Threat Models with Semantic Business Graphs

Based on research by Dinis Cruz, co-written with ChatGPT Deep Research

Executive Summary

- **Bridging Security and Business:** This document proposes **Semantic Business Graphs** that map an organization's business landscape – goals, processes, organizational structure, finances, and compliance obligations – into an interconnected knowledge graph. By linking threat modeling artifacts (risks, assets, vulnerabilities, threat actors) directly into this graph, security analyses are always performed in light of real business context. This ensures that cyber threats are evaluated not in a vacuum, but in terms of their impact on what the business cares about (e.g. customer trust, revenue, strategic goals).

- **Beyond Spreadsheets: Intrinsic Value of Contextual Graphs:** Unlike static spreadsheets or traditional BI dashboards, these semantic graphs are *living* models of the enterprise. They can answer complex, cross-disciplinary questions that siloed tools struggle with. For example, a query like *"Which critical business services would be impacted by a Log4J vulnerability, and what financial loss could that cause?"* can be answered in seconds by traversing relationships in the graph – a task nearly impossible with disconnected spreadsheets. In fact, organizations have begun using knowledge graphs to achieve **complete visibility** of their assets in business context (mapping each asset to its owner, business unit, purpose, vulnerabilities, and controls) [1]. This richer context provides insights beyond what conventional reports offer, often exceeding the capabilities of standalone BI tools.

- **Two-Way Influence between Business Decisions and Threat Models:** A semantic business graph creates a feedback loop between business strategy and security. Business decisions (like entering a new market, launching a GenAI feature, or undergoing a merger) will automatically flag relevant threat models in the graph that need updating. Conversely, findings from threat modeling (like a critical unmitigated risk on a core system) visibly link to business objectives and metrics, enabling leaders to make informed decisions (such as investing in mitigations or even pausing a product launch). This bidirectional influence means security trade-offs are always evaluated alongside business impact, fostering better risk management and strategic alignment.

- **Use Case – SaaS with GenAI (Multiple Stakeholders):** Consider a SaaS provider integrating Generative AI into its platform. For a **SaaS executive**, the graph provides a high-level dashboard of how well security risks are managed for each business goal and product feature – for instance, seeing that the "GenAI Recommendation Engine" carries two high-severity threats tied to regulatory compliance. For the **internal security team**, the graph serves as a living map of the tech stack linked to business context, letting them quickly answer questions like *"If our AI service is abused, which customer SLAs and regulatory requirements are impacted?"* For **external customers and regulators**, who often perform supply chain due diligence, the provider can extract evidence from the graph (such as mappings of controls to threats and impacted business processes) to demonstrate a mature, **context-aware security posture**. In all cases, the semantic

graph helps translate between technical risk and business value, building trust with both internal and external stakeholders.

- **Real-Time Updates via AI and Threat Modeling Teams:** To remain effective, the business graph must be kept up-to-date amid constant changes – from reorgs and acquisitions to new attack techniques. We argue that **threat modeling teams** are ideally positioned to be the custodians of these graphs, given their cross-organizational reach and mandate to understand systems deeply. By leveraging **Semantic Knowledge Graphs** and Large Language Models (LLMs), teams can maintain and query the graph in real-time. LLMs can ingest unstructured data (like architecture docs or incident reports) and propose updates to the graph, while human experts validate and refine this information. The result is a continuously evolving, version-controlled knowledge base of both business and security knowledge. As the organization changes or new threats emerge, the graph adapts in near-real-time, ensuring that security decisions are always made on the latest information. This dynamic, context-rich approach lays the groundwork for "continuous threat modeling" as a persistent, business-aligned activity rather than a one-off project deliverable.

Introduction

Threat modeling is traditionally a technical exercise: security architects enumerate assets, diagram data flows, identify threats/vulnerabilities, and recommend mitigations. Often, the output is a document or diagram for a specific system or feature. While useful, these threat models frequently live in isolation from the broader business context. Important context – like which business process relies on that system, how much revenue or customer data is at stake, or what regulatory fines could result from a breach – may not be systematically captured. This gap makes it harder to prioritize threats or communicate their significance to business leaders. In fact, many early-stage threat modeling efforts lack any business viewpoint, focusing solely on technical issues [2]. The result is that security teams might flag a "critical" vulnerability, but without linking it to business impact, it may not receive the urgency or resources it deserves.

Conversely, business decisions are sometimes made without full awareness of the security ramifications. An executive team might decide to accelerate a product launch or merge with a new company, only to later discover unaddressed security risks that accompany that move. This disconnect between business strategy and security analysis is risky: it can lead to misaligned priorities (addressing low-impact technical threats while missing high-impact business threats) and late surprises (security issues derailing business initiatives at the last minute).

Linking threat models with semantic business graphs is our solution to this disconnect. By capturing business context in a structured graph and attaching security knowledge to it, we create a common reference point for both business and security stakeholders. In this document, we expand on prior work in semantic threat modeling to describe how to construct such **Semantic Business Graphs** and use them to ensure threat modeling is always performed in the **context of real business drivers and constraints**. We will cover the structure of these graphs (the "graph of graphs" concept integrating multiple ontologies and taxonomies), how threat modeling elements map into them, and the value this provides over traditional tools. We will use SaaS companies – particularly those offering GenAI-driven services – as a running example, since they face dynamic technology and high scrutiny from customers and regulators. We also discuss how these graphs can be maintained as living artifacts, aided by AI, and why threat modeling teams are uniquely suited to curate them. The goal is to show a path forward where threat modeling evolves from a point-in-time technical assessment to a **continuous, business-aligned practice** embedded in organizational knowledge flows.

Constructing Semantic Business Graphs: Graphs of Graphs and Ontologies

To capture a comprehensive business context, a single monolithic diagram or spreadsheet won't suffice – the domain is too rich and multifaceted. Instead, we construct a **Semantic Business Graph** as a *graph of graphs*: a constellation of interlinked sub-graphs, each representing a specific facet of the organization, all connected through a unified semantic model. In practice, this means combining multiple **ontologies** (formal representations of knowledge domains) and **taxonomies** (hierarchical classifications) into one integrated knowledge graph.

Graphs of Graphs: Think of each major business domain as its own graph. For example, one sub-graph could represent **Business Strategy** – nodes for high-level goals, initiatives, key performance indicators (KPIs), and so on. Another sub-graph could model **Business Processes** – mapping how the company delivers products or services (e.g. onboarding process, payment processing workflow). Yet another graph could capture the **Organizational Structure** – departments, teams, roles, and reporting lines. Additional graphs may cover **IT Systems and Assets** (applications, databases, cloud services), **Regulatory/Compliance Landscape** (regulations, controls, standards that the business must adhere to), and **Financial Metrics** (budgets, revenue streams tied to products or units). By themselves, each of these graphs is useful. The magic happens when we link them together: an *initiative* node in the strategy graph connects to the *business process* it aims to improve; that process node links to the *IT systems* that support it; those systems link to *data assets* which have *regulatory requirements* and *owners* in the org chart, and they also link to *financial metrics* representing their contribution or cost. The outcome is a web of knowledge that mirrors the actual business – essentially an **ontology of ontologies** merged into a single navigable model.

Ontologies and Taxonomies: Underlying this graph-of-graphs is a set of ontologies defining the types of entities and relationships in each domain, and taxonomies to classify them. For example, a *Business Goal Ontology* might define that goals can be of type *Strategic Goal*, *Operational Goal*, *Sales Target*, etc., and that goals *contribute to* other goals or are *measured by* certain KPIs. A *Process Ontology* could leverage a framework like BPMN or an enterprise architecture model (such as ArchiMate) to define processes, sub-processes, inputs/outputs, and relationships to roles or systems. An *Organization Ontology* defines concepts like *Business Unit*, *Team*, *Role*, and their hierarchy. Similarly, an *IT Asset Ontology* might categorize assets into *Application*, *Database*, *Service*, *Device*, etc., and relate those to data classifications or network segments. We also include a *Risk and Security Ontology* (more on this later) for threats, controls, and impacts. Each ontology comes with a taxonomy – e.g. a controlled vocabulary of goal categories, a list of process types, a hierarchy of asset criticality levels, categories of compliance requirements (PCI, GDPR, ISO27k, etc.), and even threat categories (like STRIDE or MITRE ATT&CK tactics for classifying threats). By using formal ontologies, we ensure that each piece of data in the graph has a well-defined meaning and can be linked coherently. In essence, we are creating an **ontology of ontologies**: a higher-level structure that aligns these individual domain ontologies so they can interoperate. For instance, the ontology might specify that each *Business Process* is *owned by* a *Business Unit* (bridging process and org ontologies), or that each *IT System supports* one or more *Business Processes* (bridging IT and process ontologies). This layered modeling approach prevents the graph from turning into a chaotic tangle – it provides a **schema** and organizing principles for a very heterogeneous set of information.

Example: To make this concrete, imagine a node called **"Customer Data Analysis Service"** in the IT Systems sub-graph. According to our ontologies, this node might have the type *Application Service*. It could be connected to a **"Customer Analytics Process"** node (type: *Business Process*), indicating that this software service implements or supports that process. The process node in turn is linked to a

“Increase Customer Retention” node (type: *Business Goal*), showing that analytics is part of a strategic goal. The service node might also link to a **“Personal Data”** node (type: *Data Asset*) that it handles, which connects to a **“GDPR”** node (type: *Regulatory Requirement*), meaning the service must comply with GDPR. The service is also connected to an **“Engineering Team Alpha”** node (type: *Team* from the org ontology) as its owner, and perhaps to a **“Q4-2025 Cloud Cost = $50k”** node (type: *Financial Metric*) representing its operational cost or budget. In a traditional enterprise, each of these data points might live in a separate document or spreadsheet (one for project goals, one for process documentation, a CMDB for systems, a compliance spreadsheet for GDPR scope, an org chart for teams, a finance report for cost). By uniting them in a semantic graph, we get a holistic view: *Customer Data Analysis Service → [supports] → Customer Analytics Process → [contributes to] → Increase Customer Retention (Goal)*; *Service → [handles] → Personal Data → [subject to] → GDPR*; *Service → [owned by] → Engineering Team Alpha (Org)*; *Service → [has budget] → $50k Q4-2025* . This illustrates how graph-of-graphs integration provides **context at a glance**: one can see technical, business, compliance, and financial facets surrounding a given node. Later, we will attach threat model information to this example – e.g., linking a data breach threat to the Personal Data asset and thereby to the goal and regulation – to see the full picture.

Linking Threat Models to the Business Graph

The true power of a semantic business graph emerges when we overlay **threat modeling artifacts** onto it. In practical terms, this means representing all the key elements of threat models – such as assets, threat scenarios, vulnerabilities, threat actors, controls/mitigations, and impacts – as nodes and relationships within the same graph, and explicitly linking them to the business context nodes. By doing so, every threat or risk is **anchored in the business reality** that it affects, and every business element has visibility into its relevant security concerns.

Extending the Ontology for Security: We introduce a **Security Ontology** that integrates with the business ontologies described above. Key classes (node types) in this ontology might include: *Threat* (a potential unwanted event or attack scenario), *Vulnerability* (a weakness in a system), *Risk* (often modeled as a combination of threat, asset, impact, and likelihood), *Control* (security control or mitigation), and *Threat Actor* (the entity that might carry out a threat, e.g. hacker, insider, malware). We then define relationships like *Threat “targets” Asset*, *Vulnerability “affects” Asset*, *Threat “exploits” Vulnerability*, *Control “mitigates” Threat*, *Risk “impacts” Business Goal* (or Business Process), *Threat Actor “threatens” Asset*, etc. Many of these mappings align with existing standards – for example, a framework like MITRE ATT&CK provides a taxonomy for threat actors and techniques, OWASP Top 10 provides common threat categories for web apps, and STRIDE provides a taxonomy for threat types. Our semantic graph can incorporate those as taxonomies under the hood. The crucial part is that each security node will *connect to one or more business nodes*. For instance, an *Asset* in the security sense (like a server or database) is actually the same node as an *IT System* or *Data Asset* in the IT/business graph – we don’t duplicate it, we just ensure it has properties from both the IT ontology and the security ontology. A *Threat* node (say “Data breach of customer analytics service”) would be linked to the *Asset* (Customer Data Analysis Service) it targets, and further linked to *Business Process* (Customer Analytics Process) or *Goal* (Customer Retention) that would be impacted if the threat materialized. Similarly, a *Control* node (say “Encryption at Rest”) can link to that data asset or system and mitigate the breach threat, and it might also link to a compliance node (GDPR) if it fulfills a requirement (for example, encryption control helping with GDPR Article 32 on security of processing). By capturing these connections, we ensure that for any given threat, one can traverse the graph “upwards” to see why it matters (which goals or obligations it puts at risk), and for any given business element, one can traverse “downwards” to see how it could be harmed and what protections are in place.

Relationships and Examples: Let's extend the earlier example of the **Customer Data Analysis Service** node in the graph. Now we add security nodes: suppose there's a *Threat* node **"Unauthorized Data Extraction via API"**. This node would connect to our service node with a relationship *[threatens]→(Customer Data Analysis Service)*. Because that service handles the *Personal Data* asset, the threat also indirectly threatens that data – we could model this explicitly as *Unauthorized Extraction [threatens]→ Personal Data*. The *Personal Data* node in turn is linked to the *GDPR* requirement, so by transitive understanding we know this threat could cause non-compliance with GDPR (we might even create an explicit edge: Threat [impacts]→ GDPR, or better, Threat [impacts]→ ComplianceImpact node that has detail about penalties). Furthermore, we add a *Risk* node **"Risk of customer data breach"** which aggregates details: it might be linked to the *Threat* (data extraction), the *Asset* (personal data or the service), and have an attribute for *impact* = *High* and *likelihood* = *Medium*, etc. That risk node could link directly to the *Business Goal* "Increase Customer Retention" with a relation *[impacts]→*, since a breach would likely erode customer trust and harm retention. On the mitigation side, we might have a *Control* node **"API Rate Limiting & Monitoring"**, related to this threat. It would link to the *Threat* node with *[mitigates]→* and also link to the *Customer Data Analysis Service* (indicating this control is implemented on that system). If the control is fully in place, the risk's likelihood might be reduced; if it's not, the graph might show the risk as *unmitigated*. All these connections turn what could have been a flat list of risks in a register into a **navigable map of cause and effect**. A security analyst could click on the "Unauthorized Data Extraction" threat and immediately see: which system and data it targets, which business process and goal would suffer, which regulation is involved, who owns the system (from org chart), what control exists or is missing, and even a link to a *Threat Actor* like "Insider" or "External Hacker" if we model that. This is incredibly useful when prioritizing and discussing the threat.

Always-On Business Context: By linking threat models to business graphs, we ensure that **threat models are never considered in isolation**. For example, in a traditional approach, one might identify a vulnerability in the Customer Analysis Service and label it "High severity" based on technical impact. But now the graph provides context: this service is tied to a high-profile business goal and sensitive data. That "High" technical severity is confirmed as a *High business risk* because the impacted nodes (goal, data, compliance) are high value. Conversely, if we found a similar vulnerability on a system that was not tied to critical processes or data, the graph might reveal it's a *lower business impact* despite its technical severity – perhaps it's an internal tool with no sensitive info. In this way, the organization can focus on what really matters. The semantic graph essentially enables **risk-based threat modeling** by default: since every asset and threat is connected to impact metrics (financial, regulatory, operational), we can evaluate and query risk in a business-oriented way. Stakeholders can ask questions like *"Show me all threats that could cause over $1M in losses"* or *"List the unmitigated risks that affect our top 5 business objectives"* and get answers by traversing the graph. This is far more powerful than manually cross-referencing spreadsheets, and it ensures that security conversations with executives stay grounded in business terms.

Why Semantic Graphs Outshine Traditional BI and Spreadsheets

One might wonder: couldn't we achieve some of this with existing Business Intelligence (BI) tools or a well-structured spreadsheet? In practice, traditional tools fall short in providing the **flexibility, interconnectedness, and real-time insight** that semantic graphs offer. Here's why the graph-based approach has intrinsic value beyond what conventional BI or GRC (governance, risk, compliance) tools provide:

- **Holistic Visibility and Query Power:** A semantic business graph is like having a **living model of the entire enterprise**, rather than disparate tables or charts. BI dashboards are great at slicing and aggregating numerical data (sales figures, incident counts, etc.) along predefined

dimensions, but they struggle with *ad-hoc, relationship-driven questions*. For example, consider the query: *"Which business processes would be directly impacted if* System X *(and all its dependent components) went down due to a cyber incident, and what revenue do those processes generate per day?"* Answering that with spreadsheets or a SQL-based data warehouse would require manual joins between multiple data sources (CMDB, process documentation, revenue reports) – if those data even exist in a queryable form. In contrast, in a knowledge graph, if the relationships are in place (System X → supports → Processes; Process → has → $Revenue metric), a graph query can traverse those links and return the answer in seconds. This ability to traverse multi-hop relationships arbitrarily is a superpower of graphs. It lets you discover *indirect connections* (like a seemingly minor server failure cascading into a major business disruption because of upstream dependencies) that spreadsheets would never reveal unless someone explicitly encoded every relationship in advance.

- **Dynamic Context vs. Static Snapshots:** Spreadsheets and many BI reports are essentially snapshots – they capture a set of data at a point in time, or they update on a fixed schedule, often maintained manually. They also tend to be static in structure; adding a new type of relationship or entity might require making a new worksheet or altering the schema, which can be cumbersome. A semantic graph, however, is *flexible and extensible*. Adding a new node type or relationship type (say, modeling a new kind of risk or a new business KPI) doesn't break the existing structure; the graph simply grows to accommodate it. This means the knowledge model can evolve as the business and threat landscape evolve. For instance, if the company starts using a new category of technology (e.g. **Large Language Models**), one can extend the ontology to include an "AI Service" asset type and link it to existing processes and threats (like "prompt injection" threats) without having to overhaul dozens of spreadsheets or database tables. Additionally, the graph can be updated in near-real-time as data changes. Modern graph databases and knowledge graph platforms allow streaming updates or API-based inserts. So when a new vulnerability is discovered in System X, a script or AI agent can insert a "Vulnerability" node and link it to System X node immediately; when that system's business importance changes (say it now supports an additional process), that relationship is added. The graph is *living documentation*. This contrasts with the typical situation in many firms where documents like data flow diagrams or risk registers are updated infrequently and often lag behind reality. The value of the semantic graph is in its **timeliness and fidelity** – it's a current, accurate reflection of the enterprise and its risks, not last quarter's static report.

- **Capturing Rich Semantics (Beyond Rows and Columns):** Traditional BI operates on structured data tables. While you can model a lot in relational tables, certain complex relationships or polymorphic concepts are awkward to represent. For example, consider regulatory compliance: a single control measure (say "multifactor authentication") might mitigate multiple risks, fulfill requirements in different regulations, and apply to many systems. In a spreadsheet, you might have to duplicate entries or create a matrix to represent this many-to-many mapping. In a graph, it's natural: "MFA" node connects to all relevant systems, threats, and compliance nodes with different edge types (mitigates, implementedOn, satisfies). You preserve the *semantic distinctions* between these connections. Another example: time dimension and hypothetical scenarios. BI tools typically are backwards-looking (historic data) or current state. A semantic graph can accommodate forward-looking planning: you could insert a hypothetical node "Planned Product X" and connect threats and impacts to it, essentially doing a scenario-based threat model directly in the knowledge graph (and later either remove it or activate it when it goes live). The graph isn't constrained to numeric data either – it can hold documents, code links, personnel info, essentially any knowledge representation, all linked. It thus becomes a **unified knowledge base** for security and risk, whereas BI tools would still segment info by type (financial data in one system, incident data in another, etc., and you mentally correlate them).

- **Visual and Exploratory Analysis:** Knowledge graphs, especially when coupled with visualization tools, allow users to **explore relationships interactively**. This is not just eye-candy; it's a fundamentally different way of analyzing risk. An executive could start at a high-level node (say "Revenue Stream: European Markets") and graphically traverse outward – see which products contribute to it, then which systems underpin those products, then which threats exist on those systems. This guided exploration often yields insights (like "oh, two of our top-revenue products rely on the same third-party service – that's a concentration risk"). While BI dashboards might have drill-downs, they usually follow a predefined hierarchy and can't as easily pivot to a different connected concept. Graphs encourage **question-driven investigation**: you follow the threads of the model wherever they lead. During an incident or audit, this can be invaluable – you might uncover non-obvious impacted areas by literally following connections. For example, a knowledge graph implementation by one cyber firm allowed teams to answer in a crisis: "What has been affected? How does it impact my organization? Where should I prioritize remediation?" [3] by quickly traversing asset, business, and vulnerability nodes. Such agility in analysis is a huge value over static reports.

Figure: An example fragment of a cybersecurity knowledge graph, linking business context to technical context. In the illustration below, a single host (IT asset) is shown in context: it's linked to its business **owner** (person and business unit), its **purpose** (what application it runs or service it provides), and associated **security data** like vulnerabilities and incident tickets. By having these relationships in one view, analysts can immediately identify which assets are business-critical and see their security posture [1]. Compare this to a traditional spreadsheet where asset inventory, ownership, and vulnerabilities might be in separate files – the graph provides an integrated picture at a glance, revealing patterns and connections that spreadsheets would miss.

Source: Example knowledge graph segment (Prevalent AI)

In summary, semantic business graphs turn the typically tedious task of cross-referencing business and security data into a powerful, almost effortless query. They deliver **deep, contextual insights** that are hard to obtain otherwise. Moreover, they become an asset in themselves: a competitive advantage in how the organization understands and manages risk. Many companies already have data for business processes, org structure, CMDBs, risk registers, etc., but it's the linking of these into a knowledge graph that unlocks new value – often revealing that the whole (the connected graph) is much greater than the sum of its parts.

Bidirectional Influence: Business Decisions ↔ Threat Models

One of the most important benefits of linking threat models with business graphs is the creation of a **two-way street** between business decisions and security insights. In a traditional setup, information flow is mostly one-way: business initiatives are handed to security teams to threat-model, or security reports occasionally inform business leaders (often after a risk has materialized). By contrast, in our approach the semantic graph acts as a real-time feedback loop:

- **Business-to-Security Influence:** Every significant business decision or change can be mapped in the graph, triggering updates or reviews on the security side. For example, when the **business expands into a new region** or market, new nodes for that initiative, region, and related regulations (say data residency laws) would be added to the graph. These connect to IT systems (perhaps new deployments in that region) and data assets (personal data of new users) – which in turn flags to the security team that **new threat scenarios** must be considered (maybe region-specific threats or compliance requirements). As another example, if the company decides to

launch a new AI-driven feature (e.g., our SaaS adding a GenAI chatbot), the graph would capture a new product node linked to underlying ML models and training data assets. This immediately surfaces security questions: Does the training data include sensitive info? Could prompt injection attacks on the chatbot lead to harmful outputs? The threat modeling team, seeing this node appear, knows to engage with the product team early. Essentially, the graph makes the *latent security implications* of business moves explicit by connecting them to assets and risk nodes. It's like having a built-in mechanism that says "a new strategic move X touches these 5 systems and 3 compliance areas – security, please pay attention here." This prevents the common scenario where security is the last to know about a new project; instead, security awareness grows in tandem with the business change.

- **Security-to-Business Influence:** On the flip side, the outcomes of threat modeling and security monitoring feed directly into business decision-making through the graph. If a **critical risk** node appears in the graph (for instance, threat modeling uncovers a design flaw that could cause a major breach on a high-revenue system), that risk isn't just documented in a security report that executives might skim over. Instead, it's linked to the **business objectives, processes, and KPIs** that would be impacted. This makes the risk immediately relevant to business stakeholders. An executive viewing the dashboard sees a red flag not in technical jargon, but perhaps as "Risk of customer data breach – threatens Goal: Improve Customer Retention (potential impact: -10% retention, -$5M)". Such context-rich articulation helps business leaders appreciate the severity and allocate resources or change plans accordingly. For instance, they might decide to **delay a product launch** until that risk is mitigated, or allocate additional budget to the security team to address it, because they clearly see the connection to business impact. In effect, the graph translates security issues into the language of business outcomes, thus influencing strategic decisions. Security recommendations carry more weight when backed by business context – it's no longer just "we should fix this vulnerability" but "we should fix this because it endangers our $50M customer trust initiative".

- **Closing the Loop (Continuous Alignment):** Over time, this bidirectional flow fosters a culture of continuous alignment between security and business. Business planners will, by habit, consult the knowledge graph (or the threat modeling team maintaining it) as an early step in any major change – analogous to how they'd consult legal or compliance early on. They might ask, *"What does our risk graph say about launching in sector Y? What new threats or requirements pop up?"* Security, having an ear on changes via the graph, proactively engages rather than reactively catching up. Meanwhile, after implementation, if the threat landscape shifts (say a new vulnerability is found in a technology the business relies on), the graph's security nodes update and it's immediately clear which business services are at risk, prompting perhaps a business continuity discussion. In a sense, the graph becomes a **collaboration medium** between business and security: both contribute to it (business adds new goals/changes, security adds risks/controls) and both read from it to inform their actions. This two-way integration helps avoid the classic disconnect where the security team's priorities are different from business priorities. Here, priorities naturally sync, because a high-priority item will be one that sits at the intersection of high business value and high threat – something the graph makes readily apparent.

- **Improved Communication and Trust:** Another subtle but powerful outcome of this approach is improved communication. Non-technical executives often struggle to decipher security reports, and security teams get frustrated when their warnings go unheeded. By using the graph, discussions can revolve around a **shared model**. For instance, in a risk review meeting, instead of showing a spreadsheet of 100 risks, the security lead might bring up a visual subgraph of the top 5 risks and their connections to business elements. This could reveal, say, that **three**

different risks all impact the "Cloud Uptime" KPI for the SaaS product – a pattern that would tell the COO and CTO, "If we address these collectively, we protect our uptime promise to customers." Such visualization and clarity turn security meetings from low-understanding checkboxes into actionable strategy sessions. Business leaders start seeing security as an enabler and advisor, not just a naysayer, because the advice is delivered in business terms. Likewise, security teams gain a deeper understanding of what the business truly cares about, which improves their threat modeling focus (they will spend effort on the scenarios that matter most). Over time, this bidirectional integration builds **trust**: the business trusts that security efforts are aligned to help achieve business goals safely, and security trusts that the business will factor in their insights when charting direction.

In summary, linking threat models with semantic business graphs creates a virtuous cycle: business changes drive threat model updates, and threat insights drive business adjustments. This ensures that the organization navigates the cyber risk landscape in tandem with its business journey, with no blind spots where one outruns the other. It is a shift from security being a reactive silo to security being a proactive, integrated part of business planning and operations.

SaaS Provider Example: Perspectives from Execs, Security, and Customers

To illustrate how this approach works in a real-world setting, let's consider a **Software-as-a-Service (SaaS) provider** that heavily leverages GenAI in its product. This company offers an AI-powered platform to enterprise customers – for example, a service that analyzes customer data with machine learning to provide insights. Such a company operates in a fast-moving environment with significant intellectual property, sensitive data handling, and high expectations from customers around security and compliance. We'll examine how a semantic business-threat graph benefits three key stakeholder groups: SaaS executives, the internal security/threat modeling team, and external customers (or auditors) performing supply chain due diligence.

- **SaaS Executives (Business Leaders):** For the CEO, CTO, or other executives, the semantic graph becomes a strategic tool. On their dashboard (which queries the underlying graph), they don't just see generic metrics like "number of open vulnerabilities" – they see security metrics tied to business outcomes. For instance, the CTO might see that for each strategic product initiative, there's a risk score or heatmap derived from the graph. *Example:* The company might have a strategic goal to **"Expand into Healthcare Sector Q1 2026"**. The graph shows this goal node connected to new product features (like a healthcare-specific AI module), which in turn link to certain compliance requirements (HIPAA) and assets (a new medical data pipeline). The executive dashboard, powered by the graph, could highlight: "Goal: Expand to Healthcare – **Key Risk:** Lack of HIPAA compliance controls on new data pipeline (Risk score: High) – **Mitigation Status:** 50% controls implemented." This tells the executive at a glance that a security gap could threaten a major business objective. They can drill in (or call the security lead) and see the underlying detail – maybe the threat modeling team identified a risk of data leakage in the pipeline that would violate HIPAA, and some controls are planned but not finished. With this knowledge, the executive can make an informed decision: perhaps delay the launch until controls are complete, or allocate more engineers to accelerate the mitigation, or accept the risk formally if there's a deadline (but at least with eyes open). Without the graph, that HIPAA-related vulnerability might have been just one item in a long security report, easily overlooked or not understood in terms of its impact. Executives also use the graph to support **resource allocation and investment**. If the knowledge graph shows that a particular business unit has a cluster of high risks (maybe the AI R&D team has many experimental systems with weak security), the CFO and CTO can justify

budget to bolster security there – they can literally see how those technical risks map to potential financial or reputational damage. In board meetings, a CIO could use graph outputs to demonstrate how well security is aligned to business: e.g. "Our top 5 enterprise customers and their data flows are modeled in our graph, and I can show you we have identified and mitigated the major threats in each – thus protecting these revenue streams." This builds confidence at the leadership and board level that security is managed in a *business-driven* way, not just technically.

- **Internal Security Team / Threat Modelers:** For the security architects, analysts, and threat modelers, the semantic graph is both a **compass and a force-multiplier** in daily work. Instead of starting each threat model from scratch, they consult the existing graph to understand context. *For example,* if tasked with threat modeling a new microservice that the development team is building, they check the graph and see that this microservice will handle payment data and is part of the "Billing Process" which ties to the revenue cycle. Immediately, they know to prioritize threats around data privacy, payment fraud, and availability (since billing downtime hits revenue). They also see which controls are expected (maybe the graph shows that all payment processes should comply with PCI-DSS controls). The graph might even suggest threats: if similar systems are already in the graph with recorded threats, the team can reuse that knowledge. Furthermore, as they discover new information, they update the graph in real time. If during a design review they learn that the microservice will integrate with a third-party API, they add that as a node or property in the graph (linking that API to supply chain risk, perhaps). This means the next person who looks at that part of the business (could be compliance or another threat modeler) will immediately see the third-party connection. The graph also helps the security team **respond to incidents and questions rapidly**. Suppose a new vulnerability in an open-source library (like the infamous Log4j vulnerability) is announced that could affect their systems. Rather than scrambling through spreadsheets or contacting every dev team, the security analysts query the graph: find all software components in our product that use Log4j. Because the graph has been maintained (LLMs or scripts could even auto-annotate components with tech stack info), the team gets a list of systems, and importantly, each system comes with context of what business process or data it touches. They can then prioritize patching based on that – e.g. a system using Log4j that is internet-facing and handles customer data (high impact) is first, versus an internal tool with no sensitive info. What might take days in a large org (to identify affected assets and assess impact) can be done in hours with a well-curated knowledge graph. The threat modeling team also enjoys **cross-organizational visibility** thanks to the graph. In many companies, information is siloed – one team doesn't know what another is doing, which can lead to security gaps. But if all teams' processes and assets are reflected in the graph, the security team effectively has a map of the entire territory. They become a sort of "mission control," able to coordinate security efforts across departments using the graph to spot interdependencies. This elevates their role: they aren't just threat modelers, but **knowledge curators and advisors** to the whole business. It is worth noting that maintaining such a graph requires effort, but automation (as we discuss later) can offload a lot. The end result is that the security team moves faster and smarter – focusing on what matters, reusing knowledge, and communicating clearly – rather than being stuck in reactive firefighting or endless document writing.

- **External Customers and Supply Chain Due Diligence:** SaaS providers often face rigorous security evaluations from potential and existing customers, especially enterprise clients in regulated industries. These external stakeholders want to know that using the SaaS product won't introduce undue risk. They send questionnaires, conduct audits, ask for architecture diagrams and evidence of controls. A semantic business-threat graph becomes a **single source of truth** from which the SaaS company can pull answers for these inquiries. For example, a prospective customer's questionnaire might ask: *"Do you conduct regular threat modeling? List the*

top risks identified for your platform and associated mitigations." Without an integrated approach, answering this means hunting through documents for each product area, writing up a summary for that specific client, etc. But if the company has a maintained knowledge graph, the security team can query it for, say, all *Risk* nodes that are rated High or Critical, and retrieve their linked *Mitigations* and *Assets*. They can filter to those that pertain to the product or service the customer is buying. The result is an up-to-date list of key risks and how they're being addressed, which can be directly shared (possibly with some sanitization). This not only saves time but also demonstrates a high level of maturity. In some cases, the company might even grant a controlled view into the graph to an external auditor or customer's due diligence team – for instance, via an interactive report or dashboard generated from the graph. The external team could see, for example, a subgraph of the SaaS product they're interested in: showing components, data flows, threats, and controls. Imagine the confidence this would instill compared to a typical static Word document! It shows that the SaaS provider really understands their environment and has thought through the threats. Additionally, compliance reports for frameworks like SOC 2, ISO 27001, or industry-specific standards become easier. Those standards require mapping of controls to risks and systems – which is essentially what the graph encapsulates. Instead of assembling that mapping manually each audit cycle, the provider can export it from the graph (since *Control* nodes link to *Risk* nodes and *Assets*, one can produce a list like "Encryption Control – mitigates these 5 risks on these 3 systems – satisfies these compliance requirements" automatically). This level of organization can speed up certification processes and show auditors that security isn't ad hoc but systematically integrated with business context. Finally, from a **customer trust** perspective, being able to articulate security posture in business terms is a selling point. A sales engineer can say to a client, "We have a knowledge-driven approach to security. For every critical feature you rely on, we can show how we've threat-modeled it and protected it in line with business impact." That often resonates more than just saying "we follow best practices," because it provides evidence and context.

In this SaaS scenario, we see how the semantic graph approach scales benefits across different angles: strategic oversight for leaders, operational efficiency for security teams, and transparency for customers. It aligns everyone on a common understanding of risk in context. Particularly for companies leveraging cutting-edge tech like GenAI (which can introduce novel risks quickly), this approach ensures nothing falls through the cracks simply because "we didn't realize that piece was important." The graph makes importance explicit and helps prove to others that the company has its act together on security.

Maintaining the Graph: Keeping Pace with Change through Humans and AI

Constructing a rich business-threat knowledge graph is not a one-time effort – its value comes from being continuously updated as the organization and its environment evolve. In this section, we address how to keep the semantic graph *in sync* with reality, even as mergers happen, teams reorganize, products launch, and new threats emerge. The key idea is to leverage a combination of **process (people)** and **technology (automation and AI)** so that updates to the graph happen regularly and reliably. We'll also highlight why threat modeling teams are best suited to oversee this maintenance, acting as the human-AI orchestrators.

Integration with Change Management: To keep the graph current, it should be embedded into the organization's change management and development workflows. This means whenever there is a significant change – a new project, a major feature update, an infrastructure change, etc. – updating the graph is a formal step in the process (just like updating documentation or doing security testing might be). For example, if the company follows Agile development, adding a step in the Definition of Done for

epics that says “Update knowledge graph with new components and risks identified.” Similarly, in enterprise architecture governance, no new system goes live without being registered in the graph with its basic metadata (owner, data it handles, etc.). By tying graph updates to existing processes, we reduce the chance that things drift out of date. One practical approach is to assign **graph stewardship roles**: perhaps each agile team or product unit has a person responsible for ensuring their domain in the graph is accurate (this could be a security champion or an architect). The central threat modeling team can federate some responsibilities, receiving input from those folks for detailed changes, while they focus on consistency and linking across domains.

Handling Organizational Changes (M&A, Reorgs): Events like mergers, acquisitions, and internal reorganizations can introduce a flood of new information. For instance, in an acquisition, you suddenly have an entire new company’s worth of assets, processes, people, and risks to incorporate. Doing this manually is daunting and slow – but it’s an ideal scenario for **LLM assistance**. Imagine using an LLM to quickly ingest the acquired company’s network diagrams, asset lists, and policy documents, and having it propose a set of nodes and edges to add to the graph. For example, the LLM could read the acquisition’s documentation and output: *“System ABC – type: web app – handles customer data – maps to Business Process: Online Ordering (equivalent to acquirer’s process X) – has known risk: legacy authentication weakness.”* The threat modeling team would then review and merge these suggestions into the main graph. The ontology provides a structure to validate against (e.g., ensure new nodes use existing categories or flag if a new category is needed). This way, what might take months of manual inventory and analysis after an acquisition can be compressed into weeks or even days, with the LLM doing the heavy lifting of data extraction. Internal reorgs (say splitting a department into two, or shifting ownership of a system to a different team) are simpler – those changes can often be reflected by updating a few relationships (e.g., System X [owned by]→ Team A gets changed to [owned by]→ Team B). If HR systems or directories are integrated, some of that can be automated. Regardless, it’s important to have periodic *audits* of the graph’s accuracy, especially after big changes, to catch any stale entries.

Adapting to Threat Landscape Changes: The external threat environment is continuously changing – new vulnerabilities, new attack techniques, changes in attacker behavior, and emerging regulatory requirements. A living knowledge graph can ingest these changes as well. For vulnerabilities, integration with threat intelligence feeds or CVE databases can be set up. For example, if a new CVE is announced that affects a certain software component, and your graph has nodes for software components and their versions (or at least links systems to known libraries), an automated agent can identify matches. It could then create *Vulnerability* nodes or flags on the relevant asset nodes in the graph, linking to the CVE details. This is where an LLM can help too: it could read the description of a vulnerability and determine which assets in the graph are likely impacted (especially if names differ, etc., the LLM’s understanding can map “Log4j” to systems that have Java). Generative AI can also summarize the significance: e.g. annotate the vulnerability node with “privilege escalation risk, high severity”. Similarly, for **new attack techniques** or trends (like a novel phishing method, or a supply chain attack vector that’s trending), a security analyst could prompt an LLM with “Given our knowledge graph, which parts of our business would be susceptible to [technique]?” The LLM, if it can query or be provided a representation of the graph, might highlight relevant nodes (say “third-party dependencies” or “user account recovery process”) that align with that technique. This helps proactively adjust threat models. On the regulatory side, if a new law or standard comes out (e.g., a new data privacy law in a region), that can be added as a node in the compliance sub-graph, and an analysis (possibly AI-assisted) can find which processes or data assets involve that region’s data to link them. Essentially, the graph is not static – it’s continuously fed by both internal changes and external intelligence.

Human Oversight – The Threat Modeling Team as Custodians: While automation greatly assists, human expertise remains vital. The threat modeling (TM) team acts as the **custodians of the semantic**

graph, ensuring the quality and accuracy of the information. They define and refine the ontologies (maybe the TM lead decides to add a new category of threat after noticing a gap, or reorganize the taxonomy for clarity). They review AI-proposed changes: for instance, if an LLM suggests linking a vulnerability to a system, the team verifies it, perhaps cross-checking with the engineering team. Think of the LLM as an analyst that can comb through data and suggest connections, but the human experts validate and curate the final knowledge base – this combination yields the best results [4]. Importantly, the TM team also ensures *consistency* across the graph. With many contributors, there's a risk of duplication or slight differences (e.g., two teams might add "CustomerDB" and "Customer Database" as separate nodes referring to the same thing). The custodians can merge duplicates and enforce naming conventions or unique identifiers, often with tool support. They can also run periodic sanity checks or queries to catch anomalies (like a process node that somehow has no link to any goal or system – maybe an orphan that needs linking or removal).

Furthermore, threat modelers can use the graph to **capture lessons learned** and feed them back in. After an incident or a red-team exercise, they will update the graph with what was discovered – e.g., a previously unknown data flow or a threat scenario that was exploited. This ensures the graph gets smarter over time. The next time someone works on that area, the graph "remembers" the past issue. In essence, the TM team treats the graph as a continuously evolving *knowledge product*. They might even version-control it, as prior research suggests (storing graph data or ontology definitions in Git, for example) so changes are tracked over time [5]. This way, if something goes wrong, one can diff the graph to see what changed in the model of the world.

Real-Time Collaboration with LLMs: Maintaining a truly *real-time* graph is challenging if done purely manually. This is where LLMs integrated with the graph can offer a quasi-real-time experience. For example, an analyst can have a conversation with an AI assistant that has access to the graph: *"AI Assistant, add a node for our new microservice X, link it to the Order Processing process and tag it as containing personal data"*. The assistant, via natural language interface, could execute those instructions on the graph database. Or the analyst could say, *"I'm reviewing System Y, what do we have in the graph about it?"* and the LLM can summarize: "System Y is owned by Team Z, handles data A and B, last threat model identified 3 risks (listed), related controls are these..." This drastically reduces the friction to both read and write to the knowledge graph, moving it towards a real-time updated resource. Instead of waiting for a scheduled quarterly review to update documents, updates happen as part of daily conversations. Some forward-looking organizations are already experimenting with this combination: using chat interfaces to query knowledge graphs and even update them, making knowledge management much more interactive.

To keep up the quality, certain **automated rules** can be enforced. For example, if a node is added without an owner or without a link to a business process, the system could flag it for review – because in our model, ideally every asset should have an owner and a business purpose. These rules act like data quality guardians, implemented either in the graph database or as scripts the TM team runs.

In summary, maintaining the semantic business graph is an active, ongoing practice that blends *organizational process* (making updates part of everyone's job) with *technological augmentation* (using LLMs and integrations to automate where possible). Threat modeling teams are in the driver's seat of this effort because by nature they span silos: they talk to developers, ops, management, and external sources. They are used to translating between different views (which is exactly what the graph formalizes). With their leadership, the graph stays current and robust. The payoff is huge: when the next merger or next zero-day vulnerability comes, the organization can respond with confidence and agility, armed with a knowledge graph that is up to date and at their fingertips.

Future Outlook: Towards Continuous, Context-Aware Threat Modeling

Linking threat models with semantic business graphs isn't just an incremental improvement – it foreshadows a larger transformation in how organizations approach security and risk. As we look to the future, we can envision a world where this practice becomes the norm, leading to several notable outcomes:

- **Continuous Threat Modeling Becomes Reality:** Instead of treating threat modeling as a one-off activity during design or an annual compliance checkbox, it evolves into a **continuous process** woven into the DevSecOps pipeline. The knowledge graph, constantly fed with new data and updates, means the threat model is never stale. Every code push, infrastructure change, or business strategy update can be reflected. With automation, the system can alert when there's a drift (e.g., a new API endpoint appears that wasn't threat-modeled, or a significant business asset has no recent risk assessment). This is a shift from *periodic reviews* to *ongoing awareness*. Development and security teams could even set up CI/CD checks that query the graph, for example: if a new microservice is being deployed, the pipeline might query "does this component have a threat model entry in the graph?" and if not, flag it for immediate follow-up. The presence of the graph ensures the context for that check is readily available (it knows what "component" refers to in terms of graph nodes). Such integration makes security truly part of the lifecycle, not an afterthought.

- **Improved Decision Quality and Agility:** When executives and product managers have access to real-time risk-context information, decision-making improves. Trade-off discussions become more data-driven: "If we cut corners on Feature X to meet the deadline, the graph shows these 4 high risks will remain – is that acceptable or do we adjust scope?" These kinds of informed decisions prevent unpleasant surprises and foster a risk-aware culture at the top. Organizations become more **agile in responding to threats** too. For example, if a major incident happens to a competitor or in the industry, the leadership can instantly ask, "Do we have that exposure?" and get an answer (via the graph) grounded in facts, not guesswork. This confidence and speed can be a competitive advantage – the company that identifies and mitigates emerging risks fastest will avoid downtime and headlines. In sectors like SaaS, customers will gravitate to providers who demonstrate this level of control and foresight.

- **Security Transparency and Customer Trust:** We may reach a point where mature organizations openly share aspects of their threat models (via these business graphs) with customers and regulators to build trust. We see early signs of this with concepts like *"security transparency centers"* and detailed security whitepapers. A semantic graph could provide a sanitized but rich view that external parties can query under NDA or via interactive reports. In fact, it's not far-fetched that in the future **threat models become part of mandatory disclosures** for certain industries – akin to financial statements. Regulators might require companies to maintain an updated risk knowledge base and report key metrics from it [6] [7]. If that day comes, those who have adopted semantic threat/business graphs will be well ahead of the curve, since they'll have the data at hand to fulfill such requirements. Even if not mandated, voluntarily providing customers insights (like showing how you map their data flows to controls and threats) could become a selling point. It turns security from a black box into a value-add service ("we don't just say we secure your data – here's *how*, with evidence").

- **Industry-Wide Knowledge Sharing:** As more organizations adopt knowledge graph-based threat models, there's potential for **inter-company graph sharing** (carefully controlled, of

course). Imagine industry groups (like an ISAC – Information Sharing and Analysis Center) where common ontologies allow sharing anonymous threat nodes and patterns. For example, Company A might contribute a pattern of a supply-chain attack they experienced (with business impact annotated) to a shared repository. Company B can import that into their graph and quickly see if they have similar nodes (perhaps the same vulnerable third-party or a similar process). This network effect can significantly improve the community's resilience. Large knowledge graphs could be partially combined to form an **Internet of Risks**, where AI could run simulations – e.g., *"What if a major cloud provider goes down in region X?"* and highlight which industries or processes globally are most affected, or *"What new threats arise if everyone starts using Technology Y?"* – providing foresight at a macro level. This is speculative, but semantic standards and LLMs make it more feasible than before, because the heavy lifting of translating between different companies' terminologies can be handled by AI if the core ontologies align.

- **AI-Enhanced Risk Reasoning:** With the graph in place, more advanced AI reasoning can be applied. Graph algorithms could identify hidden risk concentrations (like many critical processes relying on one overlooked system, a classic single point of failure). They could also simulate propagation of an attack (attack graphs within the context of the business graph, combining technical attack paths with business impact nodes). Large Language Models, beyond just assisting with maintenance, could be used for "risk queries" where they combine graph data with external knowledge to answer complex questions. For instance, *"Given our current security controls and threat model, what is the risk of a ransomware attack and what would be the business impact?"* The LLM could traverse the graph for vulnerabilities, map to known ransomware TTPs, and output a narrative risk analysis citing both the internal graph and external threat intel. Essentially, the pairing of LLMs with a rich knowledge graph moves us toward **AI-assisted risk management**, where routine analysis is automated and human experts focus on strategy and decisions. The graph provides the trusted data that anchors AI reasoning (avoiding hallucinations because the AI can be instructed to stick to known graph facts when answering).

- **Organizational Empowerment and Culture:** As a softer outcome, organizations that adopt this model tend to break down silos. Since maintaining the graph requires cross-team collaboration, it forces conversations and knowledge sharing that might not happen otherwise. Application owners talk to business analysts, security talks to product managers, etc. Over time, the *culture* shifts to one of shared ownership of risk. Everyone can see how their piece connects to the whole, which often instills a greater sense of responsibility. New employees or teams ramp up faster by exploring the knowledge graph, learning not just "what do we have" but "why does it matter." And in terms of talent, having such a cutting-edge approach can attract security professionals who want to work smarter, not just grind through checklist audits. It shows the organization values innovation in security, which helps with recruitment and retention in a field where skilled people are scarce.

In conclusion, linking threat models with semantic business graphs positions organizations to navigate an increasingly complex and fast-paced risk landscape. It elevates threat modeling from a niche technical task to a **central business function** – one that actively informs and shapes the direction of the company. By continuously aligning security efforts with business reality, companies become not only more secure but also more adaptive and resilient. We move from reactive firefighting to proactive, intelligence-driven defense. In a world where digital innovation and threats grow hand-in-hand, such an integrated approach will likely become a hallmark of leading organizations. Those who embrace it early will reap benefits in clarity, agility, and trust that set them apart from competitors. The journey may require effort – building ontologies, adopting new tools, fostering collaboration – but the destination, where **every security question finds its answer in the living tapestry of the business itself**, is well worth the investment.

1 3 Harnessing Knowledge Graphs for Crisis Readiness
https://www.linkedin.com/pulse/harnessing-knowledge-graphs-crisis-readiness-prevalent-ai-rw4sc

2 simoneonsecurity.com
https://simoneonsecurity.com/wp-content/uploads/2021/03/evolving-threat-modeling.pdf

4 Threat Knowledge Graphs using Generative AI | by Venkat Pothamsetty | securitygpt | Medium
https://medium.com/securitygpt/threat-knowledge-graphs-using-generative-ai-77ede61769fe

5 How to use Semantic Knowledge Graphs for Threat Modeling | Dinis Cruz posted on the topic | LinkedIn
https://www.linkedin.com/posts/diniscruz_advancing-threat-modeling-with-semantic-knowledge-activity-7334009715133177858-5uN3

6 7 This is PART III of a series of posts containing multiple examples of the... | Dinis Cruz
https://www.linkedin.com/posts/diniscruz_this-is-part-iii-of-a-series-of-posts-containing-activity-7330166958098640897-_PgH