ChatGPT

Scaling Supply Chain Security using Threat Modeling Semantic Knowledge Graphs and Maps

Executive Summary

In today’s interconnected software ecosystem, **digital supply chain security** has become a critical challenge. Modern applications rely on countless third-party components and services, yet buyers and downstream users often lack visibility into these dependencies and associated risks [1]. This whitepaper proposes a **novel approach to scale supply chain security** by combining **threat modeling** with **semantic knowledge graphs** and visual **maps** of risk. We advocate for greater **security transparency** – envisioning a future where publishing threat models is a standard corporate disclosure (much like financial statements or ingredient labels) [2]. By representing threat models as machine-readable knowledge graphs, organizations can create “living” maps of their supply chain risks that are shareable, queryable, and continuously updated.

Key concepts are defined and illustrated, including how **semantic knowledge graphs** encode threats, assets, vulnerabilities, and mitigations in a connected ontology-driven model, and how **threat modeling maps** provide visual situational awareness of complex relationships. We discuss how linking threat models across all suppliers and components in a supply chain uncovers interdependencies, “blast radius” attack paths, and hidden single points of failure. Crucially, this approach enables **scalability, automation, and interoperability**: automated tooling (and AI) can populate and reason over the graph, while the “Graphs-of-Graphs” (G3) paradigm links multiple domains and standards into one cohesive framework [3] [4].

The paper also outlines how this strategy can integrate with existing frameworks and standards (e.g., **SLSA**, **NIST SSDF**, **ISO/IEC 27036**) by treating their security requirements as nodes in the graph, mapping organizational practices to industry benchmarks. Ultimately, **threat modeling semantic graphs** transform supply chain security from a static, fragmented checklist into a **dynamic, data-driven discipline**. Stakeholders – from security architects to executives and regulators – gain an interactive “source of truth” to assess risk, ensure compliance, and drive informed decisions. The recommendations herein blend visionary ideas with practical steps, aiming to inspire the threat modeling community to embrace semantic graphs and mandatory transparency for a safer digital supply chain.

Introduction

Modern organizations rely on **complex digital supply chains** composed of not only their own software, but also myriad open-source libraries, SaaS platforms, cloud services, and sub-suppliers. Each link in this chain – from a popular NPM package to a cloud hosting provider – introduces potential vulnerabilities that can ripple through dependent systems [5] [6]. A “simple” web application today may incorporate hundreds of indirect dependencies maintained outside the organization [7]. This complexity has real consequences: for example, the Log4j “Log4Shell” vulnerability in an obscure logging library was found in **over 100 million instances of software worldwide**, cascading a single open-source flaw into a global crisis [8]. Such incidents underscore how a weakness deep in the supply chain can propagate upward and affect countless downstream users.

Compounding the challenge, organizations often **lack visibility** beyond their immediate vendors. A company might vet its primary supplier, but that supplier could rely on fourth- or fifth-party components that remain opaque. Traditional risk management struggles to map these nested relationships [1]. Security teams are left asking: *"Where are we exposed? Which upstream component could introduce a vulnerability, and how would a breach at one supplier impact us?"* Addressing these questions requires rethinking threat modeling to encompass the **entire ecosystem**, rather than analyzing systems in isolation [9].

At the same time, **threat modeling practices** have not kept pace with the scale and speed of modern development. Threat modeling – the process of systematically identifying "what could go wrong" in a system's design – is typically done as a manual, expertise-driven exercise. The results (diagrams, spreadsheets, textual reports) are often static snapshots that **quickly become outdated** as systems evolve [10] [11]. These documents tend to be siloed within teams and lack a common format, making them hard to share or aggregate across an organization [10]. In short, current approaches **"lag behind development"** and do not scale to today's fast-paced, microservice-rich environments [12]. As one expert quipped, threat modeling is essentially an attempt to make "a fairly subjective process more objective, repeatable & consistent" – yet consistency remains elusive when models vary widely between practitioners [13] [14].

This introduction frames the dual problem: **gaining visibility into the full supply chain attack surface** and **making threat modeling scalable and continuous**. In the following sections, we present a solution that addresses both challenges: using **semantic knowledge graphs and threat modeling maps** to create an integrated, dynamic view of supply chain security. First, we discuss why **security transparency** is needed to fix the "information asymmetry" in cybersecurity markets. Then we define the key concepts – semantic knowledge graphs and threat modeling maps – and illustrate how they can transform supply chain risk management. Finally, we detail how this approach integrates with existing standards and enables automation (including the use of AI) for ongoing, real-time security assessment.

Security Transparency through Mandatory Threat Model Disclosures

A fundamental issue exacerbating supply chain risk is the **lack of security transparency** between software producers and consumers. Today, vendors typically know far more about the security of their products than their customers do. This **information asymmetry** creates a classic "market for lemons" scenario, where buyers cannot distinguish secure products from insecure ones [15] [16]. As a result, inferior security practices may go unnoticed and face little market penalty, undermining overall trust.

To correct this market failure, thought leaders have proposed making **threat model disclosures a regulatory requirement** – akin to how financial statements, safety ratings, or food ingredient labels are mandated in other industries [17]. Publishing a threat model would provide concrete evidence of the threats a company has considered and mitigated, substantiating security claims that today are often vague or unverified [18]. In the same way that transparency in finance and food safety improved those domains, **security transparency via public threat models can drive accountability and raise the baseline of security** across the software ecosystem [19].

Under this vision, organizations would regularly release standardized threat models for their products and services. Customers, investors, and regulators could then **make informed comparisons** between offerings based on these disclosures [16]. Over time, a disclosed threat model becomes an expected duty of market participation – much like no serious company today would omit publishing audited financials [20] [21]. For supply chain security, this transparency is especially crucial: consumers of software (enterprises, government agencies, etc.) could demand threat model reports from their suppliers, bringing to light the chain of assumptions and mitigations (or lack thereof) that affect downstream risk.

To illustrate, consider how **Software Bill of Materials (SBOM)** disclosures are increasingly encouraged or required, providing a list of components in a software product. SBOMs improve transparency by revealing what's inside the box. **Threat model disclosures would go a step further** – revealing how the vendor has analyzed and addressed potential abuse cases, design flaws, and dependency risks in that product. For example, a cloud service provider might publish a threat modeling "map" showing its architecture, trust boundaries, identified threats (like data breaches or supply chain attacks), and controls in place. This information, if standardized and machine-readable, could feed into customers' own risk assessments. It would also incentivize suppliers to more rigorously **assess their own dependencies** and fill gaps, knowing that omissions would be visible to the market.

Achieving this level of transparency will require industry collaboration on **formats and ontologies** for threat models, as well as safe harbor policies to encourage honest disclosure. The remainder of this paper provides a blueprint for how threat models can be captured in a **semantic, shareable form**. This not only facilitates disclosure and analysis by humans, but—importantly—enables **automation and scaling** through technology. By making threat model data **machine-readable**, we open the door to powerful tools that can integrate, visualize, and reason about supply chain risks end-to-end.

Semantic Knowledge Graphs: A Foundation for Scalable Threat Modeling

To enable scalable and interoperable threat modeling, we turn to **semantic knowledge graphs** as the core representation. A **semantic knowledge graph** is a way of structuring information as a network of entities (nodes) and relationships (edges) with explicit meaning defined by a schema or ontology [22]. In a security context, relevant entities might include *Systems, Assets, Vulnerabilities, Threats, Actors, Controls,* and so on – each represented as a node. Relationships capture how these entities interact or relate: e.g., *"Asset* has *Vulnerability"*, *"Threat* targets *Asset"*, *"Control* mitigates *Threat"*, *"Actor* exploits *Vulnerability"*, etc. [23] [24]. What makes the graph **semantic** is that every node and edge follows a formal ontology that gives it well-defined meaning (often aligning with industry standards or taxonomies) [22]. This shared vocabulary ensures that both humans and machines interpret the data consistently.

By modeling threat models as rich, connected graphs of data, we elevate them to **first-class, shareable artifacts**. Unlike static Word documents or spreadsheets, a graph-based threat model is **queryable** and **mergeable**. Multiple teams can contribute to a unified knowledge graph, and their contributions won't conflict as long as they adhere to the schema. For example, one team might add nodes for a new microservice and its threats, while another adds nodes for a new vulnerability; the graph naturally links these if the vulnerability affects that microservice. **Machine reasoning** can be applied on top: automated queries or inference rules can traverse the graph to identify risks and gaps that would be hard to spot manually. If we want to find all applications lacking encryption on sensitive data flows, we can query the graph for all *"Data Asset"* nodes labeled sensitive that have no *"mitigated-by: EncryptionControl"* edge. This is far faster and more reliable than hunting through documents or spreadsheets for each system [25] [26]. The moment someone updates the graph (say, adds a missing control or a new threat), that change propagates through all analyses and views automatically [27].

Crucially, semantic graphs enable **contextual enrichment** from external sources [28]. Because the schema is flexible and based on ontologies, we can **plug in data feeds** directly. For example, an external vulnerability database (CVE feed) can be linked into the graph: if a new CVE is published affecting an open-source library in our inventory, a *"vulnerability"* node for that CVE can link to the *"Library"* node in the graph, instantly flagging an issue [28] [29]. Similarly, real-world incident data or threat intelligence can be mapped onto our threat scenarios – if an attack that matches a scenario in our model has occurred in the wild, we can link an *"Incident"* node to that *"Threat"* node. Compliance requirements or controls from frameworks (ISO, NIST, OWASP, etc.) can be represented as nodes too [29]. All these linkages turn the threat model into a **rich, multi-dimensional map of risk** [29], where technical, business, and compliance views all intersect in one model.

In summary, using semantic knowledge graphs for threat modeling transforms it from a one-time, subjective exercise into an **ongoing, data-driven practice** [30]. The knowledge graph acts as a living repository of security knowledge that can grow and evolve with the system. It provides a **"single source of truth"** that is both machine-readable and human-explainable [31]. In the next sections, we will explore how these graphs serve as the basis for **threat modeling maps** that visualize security relationships, and how they can be applied specifically to model and secure the software supply chain.

Threat Modeling Maps: Visualizing Relationships and Risk

While a semantic knowledge graph provides the back-end representation of security knowledge, it is equally important to consider how this information is presented to humans. This is where **threat modeling maps** come in – the visual counterpart to the knowledge graph that helps practitioners and stakeholders intuitively understand the threat landscape. By "map," we refer to any diagram or visualization that is generated from the knowledge graph to show entities as nodes and their relationships as connecting lines. These maps allow us to literally **see** the structure of the system and its risks, supporting better decision-making.

In a supply chain context, a threat modeling map might depict your organization and all its key vendors, software components, and data flows as an interconnected web. Nodes could be color-coded by type (e.g., internal system, third-party service, open-source library, data asset, etc.), and edges labeled by relationship (e.g., *uses*, *depends on*, *hosts*, *exposes data to*, *is vulnerable to*). Security information overlays can be added: for instance, nodes outlined in red could indicate a high-risk vendor or a component with a known critical vulnerability; padlock icons might denote where important controls (like encryption or MFA) are in place. Such a map becomes a high-level **situational awareness tool** – a living blueprint of the supply chain's security posture [32] [33].

Dinis Cruz often emphasizes evolving graphs into **"maps that enable situational awareness"**, underscoring that how we present data is as important as the data itself [34]. A well-designed threat model map allows both technical experts and non-technical stakeholders (like executives or partners) to grasp complex relationships at a glance. For example, a map could quickly highlight that *Supplier X* has direct connections to your crown jewel database (via an API), or that two seemingly unrelated business services actually share a dependence on the same third-party authentication provider – an **unseen common point of failure**. These insights can easily be missed in text-based reports but **"one glance can show that a single point of failure spans many systems"** when visualized on a graph [35].

Another advantage of mapping is to support **"blast radius" analysis** for potential attacks. By tracing paths on the map, an analyst can determine how a threat could propagate. For instance, an attacker compromising a lower-tier supplier might traverse through an integration to reach a primary vendor, and from there into your system – the map would show this multi-hop path clearly via connected edges. As cited in one study, if an open-source component is used by multiple applications, a vulnerability in that component radiates impact to all those applications; on the graph, all those app nodes link to the vulnerability node, visibly expanding the influence scope [36] [37]. This "long chain" of dependencies is naturally mirrored by the graph model [38] [39], whereas in a spreadsheet it would be very difficult to follow.

To make these maps practical, organizations should leverage graph visualization tools that can plug into the knowledge graph. The tools should allow filtering and layering of information – e.g., view the whole ecosystem or zoom into one supplier's subgraph, toggle overlays for compliance controls or incident history, etc. [40] [41]. Interactive features (hover for details on a node, click to see associated threats/ controls) turn the threat model into a navigable risk register. By using a semantic graph as the data source, every element on the map is backed by structured data, meaning you can drill down to as much detail as needed. Ultimately, **threat modeling maps** derived from semantic graphs become "living" diagrams that update as the underlying data changes. They provide an accessible way to communicate complex security concepts, helping bridge gaps between technical risk analysis and executive-level understanding.

Threat Modeling the Entire Supply Chain

Traditional threat modeling typically focuses on one system or application at a time. In contrast, securing the supply chain demands that we **extend threat modeling beyond organizational boundaries** [42]. The key insight is that *every supplier, third-party service, or software component can be treated as an entity with its own threat model* [43]. In other words, we perform threat modeling **recursively** at each node in the supply chain and then aggregate those findings to understand systemic risk [43] [44].

Practically, this means when you model threats to your system, you must include not only internal design flaws but also threats arising from dependencies **upstream** and **downstream**. For example, if your web application relies on a third-party payment API, your threat model should account for the risk of that API being compromised or unavailable [45]. Likewise, if that payment provider in turn relies on a cloud infrastructure service, then the cloud provider's outages or breaches become part of *your* threat landscape [46]. By viewing vendors and sub-vendors as integral parts of your attack surface, you create a **hierarchy of threat models**: your organization at the top, underpinned by threat models of each critical vendor, which include models of *their* dependencies, and so forth [47]. Any weakness in a lower tier can cascade upward. As the adage goes, your security is only as strong as the weakest link in your vendor ecosystem [48].

To operationalize this, organizations can define **threat scenarios at multiple levels**: e.g. *"What if Supplier X's user database is breached?"*, *"What if open-source library Y has a zero-day vulnerability?"* [49]. Each of these scenarios is essentially a mini threat model for that component, identifying potential adversaries, entry points, and mitigations. Industry resources can help inform these scenarios – for instance, MITRE's catalog of supply chain attack patterns or known incidents like malicious package injections can serve as templates [49]. By cataloguing such threat models for key third-party and fourth-party elements, you build a **knowledge base of risks spanning the entire supply network** [50].

The challenge is how to manage and make sense of this sprawling collection of threat data. Manually keeping track of all the interconnections (who uses what, which vulnerability affects which component, which supplier has access to which data) **quickly becomes infeasible** with traditional tools [51]. This is precisely where **semantic graph representations become invaluable** [52]. By encoding each component's threat model into a shared knowledge graph, we can **link them via dependency relationships** to form an integrated picture. Each individual system or supplier's threat model exists as a *subgraph* (containing that system's assets, threats, and controls), and the supply chain knowledge graph then links these subgraphs together by their connections (e.g., *System A depends on Library B*;

Vendor X provides Service Y to Company Z) [53] . In effect, we get a **"graph of graphs"** where the entire supply chain's risk structure is captured in one connected model [54] .

This aggregated view exposes **systemic risks** that might be invisible when looking at one supplier in isolation. For example, the graph may reveal that two critical vendors you rely on actually depend on the *same* third-party cloud service – an *"unseen common dependency"* that represents a single point of failure for your business [55] [40] . Or you might discover that a particular open-source component is used in 20 different applications across your organization; thus a single vulnerability in that component simultaneously threatens all those applications [36] [56] . The knowledge graph, by mirroring the actual dependency network, makes such **non-obvious linkages** apparent.

By connecting threat models across the supply chain, we form a **holistic meta-model of the entire ecosystem** [57] . Analysts can zoom out to see an overarching map of the supply chain and then zoom in on any specific node's details as needed [58] . Thanks to the semantic ontology underpinning the graph, relationships and entity types are consistent across all tiers, so comparisons and automated reasoning hold true globally. For instance, a *"data breach"* threat in a SaaS vendor's model can link to a *"Data Confidentiality Impact"* node in your own model, aligning terminology so that risk severities and mitigations can be evaluated coherently across organizational boundaries [59] .

Graphs-of-Graphs (G3): Enabling Scalability and Interoperability

As the supply chain example shows, linking multiple threat model graphs together yields powerful insights. This approach is generalized in the concept of **Graphs of Graphs of Graphs (G3)** – an emerging paradigm for scalability in threat modeling and security knowledge management. **G3 refers to connecting multiple knowledge graphs across different layers or domains into a larger, federated graph-of-graphs structure** [3] . Instead of trying to force everything into one monolithic mega-graph, we maintain modular subgraphs for distinct concerns and then link them through common reference points.

In practice, each subgraph might represent a particular domain of security knowledge. For example, one graph could capture your internal application architecture and threats; another graph could represent an industry threat intelligence database (attack techniques, actor groups, etc.); yet another could model compliance controls from a standard like NIST or ISO. With G3, these graphs can be **interlinked**. Each subgraph becomes effectively a node within a higher-level graph. This is done by establishing **shared ontologies and identifiers** so that entities in different graphs can reference each other. As Cruz's work on Semantic OWASP illustrates, we can treat community knowledge bases (like the OWASP Top 10 list of vulnerabilities or MITRE ATT&CK tactics) as *semantic data* that our internal graphs link to [60] [61] . For instance, a node *"SQL Injection"* in your threat model graph might link to a canonical *"SQL Injection"* node in a global OWASP ontology graph [62] [61] . This ensures consistency (everyone is referring to the same concept) and allows knowledge to be shared.

The G3 approach brings significant **interoperability** benefits. It means that when different teams or tools produce graphs (e.g., one team models cloud infrastructure risks while another models software supply chain risks), those graphs can interconnect into a unified meta-graph. Under the hood, this requires aligning the schemas (ontologies) used by each graph so that common concepts map to each other [63] [64] . Using standards from the semantic web (RDF/OWL) or similar, one can formally define ontologies for the security domain that cover threats, vulnerabilities, assets, controls, etc. – providing the "language" that all graphs speak [65] [64] . If one model calls something "Privilege Escalation" and another calls it "Elevation of Privilege", they should both link to a unified concept defined in an ontology.

G3 ensures that **data from different sources can be merged and reasoned over seamlessly**, eliminating fragmentation.

Scalability is another major advantage. With graphs-of-graphs, each subgraph can be maintained and scaled independently by the subject matter experts or automated systems responsible for it. There's no need for a single gargantuan database that knows everything. Instead, references (edges between graphs) keep the knowledge connected. This modular approach mirrors the microservices philosophy in software architecture – it's easier to manage many small components than a single huge one. When combined, they act as one system. As our threat modeling needs grow (covering more systems, more suppliers, more types of threats), G3 allows the knowledge base to **scale out** without collapsing under its own weight.

Finally, G3 aids **collaboration and industry-wide knowledge sharing**. Different organizations or communities could publish portions of their threat knowledge as graphs, and consumers could incorporate those by reference. Imagine an open repository of threat models or control libraries that your company's graph can plug into, rather than reinventing the wheel. The ultimate vision is an ecosystem of interoperable security graphs, where data from vendors, users, researchers, and regulators can all link together. In essence, G3 turns isolated efforts into a **collective intelligence network** for cybersecurity. The next section will explore how we can leverage this approach to integrate well-known security frameworks into our model, ensuring that our supply chain threat modeling aligns with best practices and standards.

Integrating Industry Frameworks via Graph Links

Any robust security program will draw on established frameworks, best practices, and standards. Examples in the supply chain security context include **SLSA (Supply-chain Levels for Software Artifacts)**, which defines maturity levels for software build integrity; **NIST's Secure Software Development Framework (SSDF)**, which provides guidelines for secure development lifecycle; and **ISO/ IEC 27036**, which gives guidance on supplier relationship security. A key benefit of using semantic graphs for threat modeling is the ability to **integrate these frameworks directly into the model** – essentially creating graph-based mappings between your organization's security data and the frameworks' requirements.

In a knowledge graph, we can represent the elements of a framework as nodes and relationships, just like any other data. For instance, consider **SLSA's levels**: SLSA Level 4 (the highest) has certain criteria like hermetic builds and two-person code reviews. Each of these criteria can be modeled as a *"Control"* or *"Requirement"* node in an ontology. Your supply chain graph could then link those nodes to the parts of your system that meet (or don't meet) the criteria. If your build pipeline node lacks the *"hermetic_build"* property, a query can flag that you haven't met SLSA Level 4. Similarly for NIST SSDF: its practices (e.g., "Implement threat modeling" or "Verify third-party components") can be nodes in a *Compliance* subgraph, and you can connect evidence of implementing those practices in your environment to those nodes.

Dinis Cruz's approach suggests treating compliance frameworks as simply another **layer in the graph** [29]. For example, you might import the control catalog of NIST 800-53 or ISO 27001 as a set of nodes (organized by families/categories) in your knowledge graph. Then, link each control to the relevant threat or asset nodes in your model that implement that control or address that risk. In effect, this creates a *traceability matrix* automatically: you can trace how each compliance requirement is fulfilled by specific technical measures, and conversely, see which controls map to which threats. In the words of Cruz, **"compliance frameworks (like ISO 27001 controls or OWASP ASVS requirements) can be**

represented as another layer in the graph, ensuring that your threat mitigations cover required controls” [29]. This approach not only helps during audits and assessments (you can generate compliance reports by querying the graph), but also ensures that security and compliance efforts are not siloed. Everything converges in one model.

As a concrete illustration, consider **ISO/IEC 27036**, which focuses on supplier security. It might recommend policies like “assess supplier’s security capabilities” or controls like “include security requirements in supplier contracts.” In our semantic graph, we could have an ontology class for *ComplianceRequirement* and add nodes for each key ISO 27036 control. One such node might be *“Supplier Security Assessment Conducted”*. We could then link that node to our *Supplier* node (e.g., Supplier X) with a relationship like *compliant_with* if indeed we have evidence that an assessment was done. If the link is missing for a critical supplier, that gap is evident on the graph (perhaps showing Supplier X in a different color or with an “unassessed” status). In this way, the graph serves as a **dynamic compliance dashboard**, highlighting where you stand against frameworks across all your dependencies.

Similarly, for **SLSA**, we can map the entire software pipeline: nodes representing source repositories, build systems, artifact registries, etc., each annotated or connected with SLSA compliance nodes (e.g., *“cryptographic provenance attested”*). The knowledge graph could answer a query like “List all production applications that are not built under SLSA Level 3 or higher” by traversing these links. **NIST SSDF** practices (e.g., vulnerability management, secure code training) could be linked to nodes representing teams, processes or tools in the organization responsible for those practices.

By integrating frameworks into the threat modeling graph, we achieve two outcomes: **1) Comprehensive coverage** – making sure that for every framework control there is a corresponding consideration in our model (or explicitly marking it not applicable), and **2) Simplified updates** – if a framework updates or if we adopt a new standard, we can map it into the graph and immediately see where we need to improve. This approach aligns with the idea of “From Static Standards to Semantic Graphs” – turning textual standards into living data that can interface directly with your environment [66] [67]. It bridges the gap between high-level best practices and on-the-ground security architecture. In essence, our knowledge graph becomes a **universal adapter** that links threats, mitigations, and controls in our enterprise to the language of industry standards and regulations.

Automation and Continuous Risk Monitoring

One of the most powerful aspects of using semantic graphs and maps for threat modeling is the potential for **automation and continuous monitoring**. A major criticism of classical threat modeling is that it’s labor-intensive and often point-in-time. Here, we outline how automation – including **Graph Analytics** and **AI/ML (e.g., LLMs)** – can keep the graph updated and derive insights in real-time, turning our supply chain threat model into a living, proactive defense tool.

Data Ingestion and Graph Updates: Automation should start with feeding the graph fresh data. An important first step is building an **asset and dependency inventory**. Tools can automatically generate a Software Bill of Materials for each application (listing all open-source libraries and versions) and push that into the graph [68] [69]. Integrations with CI/CD pipelines and cloud management APIs can update the graph whenever new components are introduced. Likewise, a continuous feed of vulnerability data (from sources like NVD for CVEs, or proprietary scanners) should update *Vulnerability* nodes and link them to any affected *Component* nodes [70] [71]. Vendor management systems and security questionnaires can be another feed – for example, if a supplier attests to having ISO 27001 certification, a *Compliance* node for that can be added/updated. All these updates ideally occur on a schedule or trigger (daily, or immediately when new code is deployed, etc.), so the graph is never stale. As Cruz notes, **automation is crucial** to continuously pull data from scanners, questionnaires, and intel feeds to keep the model current [72].

Automated Reasoning and Alerting: With data continuously flowing in, we can implement rules and analytics to interpret it. A simple example is to automatically flag combinations of conditions that represent high risk. The knowledge graph allows encoding **decision logic** either in code or even as additional nodes/relationships. For example, we might formalize a rule: *"If a critical vulnerability (CVSS > 9) exists in a library used by a critical application, and no patch is available, then mark that application's risk as High."* This rule can be run as a query on the graph, or manifested as a separate *Risk* node linked to the application node, etc. [73] [74]. We could maintain a library of such rules corresponding to our risk appetite. More advanced graph algorithms (like centrality measures) can identify which nodes are most connected and might represent concentration risk (e.g., a single library used everywhere). The **"blast radius"** queries described earlier are another automated analysis: if a node representing a component gets a new vulnerability, the graph can instantly list all upstream systems impacted [36] [37]. These analyses can be integrated with alerting systems – for instance, a critical change in the graph (like a new zero-day in a heavily used component) could automatically generate an alert or a ticket in a tracking system [75] [76].

AI and LLM Integration: Recent advancements in AI, particularly large language models, can supercharge the graph-based approach. LLMs are very good at making connections and suggestions when given structured knowledge. One way to use them is to enable **natural language querying** of the threat model. A security analyst (or even a manager) could ask in plain English: *"Which of our suppliers lack multi-factor authentication and could impact customer data?"* The LLM can translate this question into a graph query (by understanding that we're looking for *Supplier* nodes that are connected to *Customer Data* nodes but have no *MFA control* linked) and then answer with the relevant suppliers [77] [78]. The model can even provide the reasoning path (increasing trust in the answer) because it can cite the graph relationships it traversed [79].

Another use of AI is in **maintaining and expanding the graph**. For example, if we ingest a new policy document or a security questionnaire, an LLM could parse it and suggest updates to the knowledge graph (like adding a new *ComplianceRequirement* node for a new policy clause, or flagging that a supplier's answer indicates a certain risk). Cruz's vision in Project SupplyShield involves GenAI working alongside human analysts to populate and interpret the graph [77] [80]. Importantly, because the graph provides structure and provenance, even an AI-driven suggestion can be verified and traced – unlike a black-box AI output, the graph's edges provide a transparent chain of reasoning [77] [81].

Continuous Feedback Loop: With these automation pieces in place, we effectively create a continuous feedback loop for supply chain security. As soon as something changes in our environment or threat landscape – a new dependency, a new vulnerability, a new supplier – it is reflected in the graph. The analytics then re-evaluate risk in light of that change, and if needed, trigger alerts or updates to threat models. Conversely, if mitigations are added (say a patch is applied or a compensating control is deployed), the graph is updated and the risk scores decrease accordingly. This approach turns the threat model into a **"living, breathing" risk model that is always current** [82] [83]. In essence, we shift from periodic, static assessments to **continuous monitoring**. Security teams can focus on investigating the insights and warnings surfaced by the system rather than manually collecting data.

By leveraging automation and AI in tandem with semantic threat modeling, we finally get closer to making **"continuous threat modeling"** a reality [84] [85]. This not only improves security posture day-to-day but also makes the process more **repeatable, explainable, and scalable** (since much of the heavy lifting is done by tools). All actions and decisions remain grounded in the graph, which means they are auditable and based on evidence, addressing one of the key concerns with AI in security – the need for trust and provenance. In conclusion, automation transforms our semantic knowledge graph from a static repository into an active defense and decision-support system for supply chain security.

Conclusion

The increasing frequency of supply chain attacks and the deep complexity of modern software ecosystems call for a paradigm shift in how we approach security. In this whitepaper, we presented **Threat Modeling Semantic Knowledge Graphs and Maps** as a foundation for scaling supply chain security. By making threat models **mandatory disclosures**, we introduce much-needed transparency into the software marketplace, empowering stakeholders to make risk-informed choices and incenting providers to up their security game [16] [19]. By leveraging **semantic knowledge graphs**, we transform those threat models from static documents into **living knowledge bases** that can be analyzed, shared, and updated continuously [86] [85]. And by visualizing the data as **threat modeling maps**, we gain human-friendly views that mirror the real-world complexity of our digital supply chains, providing a true "map of risk" for situational awareness [87].

A recurring theme of this paper is **integration** – integration of data, of processes, and of standards. We showed how a graph-of-graphs (G3) approach allows disparate sources of security knowledge (from internal systems to industry frameworks) to interconnect into a unified model [3] [4]. This interoperability is crucial for tackling supply chain issues that no single organization can fully understand in isolation. We also discussed how aligning with frameworks like SLSA, NIST SSDF, and ISO 27036 can be achieved by representing their controls in the graph, ensuring that our security investments are both comprehensive and compliant. In essence, the semantic graph becomes a **backbone for collaboration**, not only within a company (between developers, security, compliance, and executives) but also across the community (imagine open threat model exchanges and standard ontologies that everyone contributes to).

Scalability and automation were emphasized through the use of AI and continuous monitoring. The approach outlined is inherently **future-proof**: as new threats emerge, new technologies arise (e.g., the AI supply chain itself becoming a target), or new regulations come into play, our graph-based model can flex and extend to accommodate them [88] [89]. Graphs, by nature, are extensible – we can always add new node types or relationships without starting from scratch [88] [90]. This agility means the security program can adapt in near-real-time to the ever-shifting landscape. The "living" threat model, powered by automation, ensures that security is not a one-off project but a continuous practice woven into the fabric of development and supplier management.

For the threat modeling community and industry at large, the implications are significant. We should invest in developing **common ontologies and schemas** for security graphs, so that when threat models are shared as disclosures, they can be readily consumed by tools and compared across vendors. Regulators and standards bodies could accelerate this by specifying formats for threat model reporting (similar to how financial XBRL works, or how NIST's OSCAL provides formats for control catalogs [66]). Tool vendors should incorporate graph databases and visualization into next-generation GRC and threat modeling products. And organizations can start small – perhaps by choosing a pilot system or critical supplier – to build a mini knowledge graph and threat model map, experiencing firsthand the clarity it brings.

In closing, **scaling supply chain security using threat modeling semantic knowledge graphs and maps** is about marrying the power of data science with the wisdom of security engineering. It turns tacit knowledge into explicit models, isolated checklists into connected maps, and reactive audits into proactive continuous assurance. By doing so, it promises a future where supply chain attacks are not mysterious “black swan” events, but risks we actively track, manage, and mitigate through collective transparency and intelligence. The path forward is challenging but attainable – and the cost of not pursuing it is an ever-growing risk hidden in the tangled webs of our digital supply chains. Let us move towards a world where every link in the chain is illuminated by knowledge and every software build comes with a **semantic threat model** that is open, standardized, and actionable.

---

1 4 5 6 7 8 9 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 87 88 89 90 Using Threat Modeling and Semantic Graphs to Secure the Digital Supply Chain - Dinis Cruz - Documents and Research
https://docs.diniscruz.ai/2025/05/30/using-threat-modeling-and-semantic-graphs-to-secure-the-digital-supply-chain.html

2 15 16 17 18 19 20 21 Threat Models as Mandatory Disclosures: A Vision for Security Transparency
https://files.diniscruz.ai/github/pdf/2025/05/29/threat-models-as-mandatory-disclosures__a-vision-for-security-transparency.pdf

3 31 60 61 62 63 64 65 Graphs of Graphs of Graphs (G3) in Threat Modeling - Dinis Cruz - Documents and Research
https://docs.diniscruz.ai/2025/05/30/graphs-of-graphs-of-graphs-g3-in-threat-modeling.html

10 11 12 13 14 22 23 24 25 26 27 28 29 30 84 85 86 Advancing Threat Modeling with Semantic Knowledge Graphs - Dinis Cruz - Documents and Research
https://docs.diniscruz.ai/2025/05/29/advancing-threat-modeling-with-semantic-knowledge-graphs.html

66 67 Scaling Europe’s Regulatory Superpower: From Static Cybersecurity Standards to Semantic Graphs - Dinis Cruz - Documents and Research
https://docs.diniscruz.ai/2025/03/31/scaling-europe-regulatory-superpower.html