GRAPHSEC: Graph-Based Supply Chain Attack Detection and Risk Propagation Analysis for Enterprise Salesforce Deployment Pipelines

Lalith Chandra Bandaru[1], Mohammed Shakeer Bandrevu[2]

[1,2]Independent Researcher

Abstract:
Software supply chain attacks targeting enterprise deployment pipelines have escalated dramatically in frequency and sophistication over the period 2020 to 2023, with the Salesforce ecosystem facing a particularly complex supply chain attack surface encompassing managed package distribution channels, CI/CD pipeline infrastructure, third-party API integrations, and the credential chains connecting development environments to production CRM systems. Existing supply chain security tools are designed for traditional software ecosystems (npm, Maven, PyPI) and do not model the Salesforce-specific dependency relationships between managed packages, deployment service accounts, metadata components, and runtime integration endpoints. We introduce GRAPHSEC, a graph-based supply chain security framework that models the complete Salesforce deployment pipeline as a weighted directed graph in which nodes represent software components, credentials, infrastructure elements, and data flows, and edges represent dependency, trust, and data transfer relationships with associated compromise probability weights. GRAPHSEC applies graph-theoretic risk propagation to compute component-level compromise risk scores that reflect both direct vulnerability exposure and transitive risk inherited through supply chain dependencies. The framework integrates with the Secure CI/CD framework to provide real-time attack path analysis and alert generation. Evaluated across eleven enterprise Salesforce deployments over fifteen months, GRAPHSEC achieved 95.9% weighted detection precision across five attack categories, reduced mean time to detect supply chain incidents from 34 days to 2.1 days, and identified 73 high-risk attack paths that deployers were unaware of before GRAPHSEC deployment.

Keywords: supply chain security, graph-based risk analysis, Salesforce, attack path analysis, dependency confusion, managed packages, CI/CD security, risk propagation, compromise probability.

1. INTRODUCTION

The security of software supply chains has emerged as one of the defining enterprise security challenges of the early 2020s. The SolarWinds attack of December 2020 demonstrated that nation-state adversaries are capable of compromising the software build and distribution systems of major technology vendors, injecting malicious code into widely deployed products before they are distributed to customer organisations. Subsequent research identified more than 700 open-source supply chain attacks in 2022 alone, representing a 633% increase over the 2019 baseline. Enterprise Salesforce deployments are embedded in software supply chains of considerable complexity: they depend on AppExchange managed packages whose source code organisations cannot independently inspect, CI/CD pipelines that execute with deployment permissions on production CRM systems, third-party API integration endpoints whose certificates and code can change without notice, and credential chains that link developer workstations through version control systems and build infrastructure to Salesforce production orgs. Any element in this chain represents a potential entry point for supply chain attack.

The distinguishing characteristic of supply chain attacks that makes them particularly difficult to defend against is their exploitation of trust relationships rather than technical vulnerabilities. A managed package installed from AppExchange is trusted by the Salesforce platform to execute with full org permissions because the package has passed the AppExchange security review — a point-in-time assessment that may not reflect the security posture of subsequent package updates. A CI/CD runner that executes deployment scripts is

trusted by the pipeline to access production deployment credentials because it has authenticated through the configured service account mechanism — trust that is exploited when an attacker compromises the runner infrastructure. The challenge for supply chain security is therefore not simply detecting vulnerabilities in individual components but modelling the trust relationships between components and the paths through which a compromise of one component propagates risk to others. This is inherently a graph analysis problem, motivating the GRAPHSEC approach.

Existing supply chain security tools address the Salesforce deployment ecosystem inadequately. Software composition analysis tools designed for npm, Maven, and PyPI package ecosystems have no coverage of Salesforce managed packages, which are distributed through AppExchange under a proprietary distribution model with no public package vulnerability database. Pipeline security scanners that focus on CI/CD configuration files do not model the trust relationships between pipeline components and the production systems they access. The LTDF threat detection framework [12] provides excellent runtime anomaly detection for Salesforce user behaviour but does not model the supply chain relationships that determine which components a runtime anomaly might propagate to. The secure CI/CD framework [11] provides pre-deployment security controls but does not model attack propagation across the multi-component supply chain.

GRAPHSEC fills these gaps through three contributions worth naming up front. First, a Salesforce-specific supply chain graph model that captures the full set of dependency, trust, and data flow relationships relevant to enterprise Salesforce deployments, extending beyond the package dependency graph of traditional SCA tools to include infrastructure trust relationships, credential dependency chains, and runtime integration connectivity. Second, a graph-theoretic risk propagation algorithm that computes component-level compromise risk scores by aggregating direct vulnerability scores with the weighted transitive risk propagated through the graph from other at-risk components. Third, an attack path analysis module that identifies the highest-risk paths through the supply chain graph — the sequences of component compromises that would give an attacker the most valuable access with the lowest combined compromise probability — and surfaces these paths as actionable security findings for remediation.

The prior works in this research programme establish the individual component security controls that GRAPHSEC synthesises into a unified risk model. The secure CI/CD framework [11] provides per-component security controls at the pipeline stage level; the URGF governance framework provides deployment-time policy enforcement; the LTDF runtime monitoring framework [12] provides production anomaly detection. GRAPHSEC adds the cross-component risk propagation model that connects these individual controls into a unified supply chain security posture, enabling security teams to understand not just the risk of individual components in isolation but the aggregate risk of the interconnected supply chain as a system.

2. BACKGROUND AND RELATED WORK

2.1 Supply Chain Attack Models

The academic study of software supply chain security has accelerated rapidly since 2020. Ohm et al. [1] present a comprehensive dataset of 174 malicious packages published to npm, PyPI, and RubyGems between 2015 and 2019, identifying five attack techniques: typosquatting (publishing packages with names similar to popular packages), dependency confusion (publishing public packages with names used only in private registries), code injection into legitimate packages through compromised maintainer accounts, malicious code introduced through pull request merges, and StarJacking (linking packages to unrelated popular repositories to appear trustworthy). Of these five techniques, dependency confusion and maintainer account compromise are directly applicable to the Salesforce managed package ecosystem; the other three apply primarily to the npm and Python package dependencies used in CI/CD pipeline scripts. Ladisa et al. [2] extend this analysis with a systematic taxonomy and threat model that GRAPHSEC uses as the foundation for its attack category definitions.

Graph-based approaches to supply chain security have been proposed in the academic literature but have not previously been applied to the Salesforce-specific deployment context. Sejfia and Schäfer [3] propose a package dependency graph model for detecting anomalous dependency chains in npm ecosystems. Steenhoek et al. [4] conduct an empirical study of deep learning models for vulnerability detection, evaluating nine state-of-the-art models and identifying key limitations in model robustness and training data composition. GRAPHSEC extends these approaches in three important directions: the graph includes non-package nodes

(infrastructure, credentials, runtime endpoints) that are absent from previous package-centric models; the edge weights represent empirically calibrated compromise probabilities rather than binary relationships; and the risk propagation algorithm is adapted for the specific trust semantics of the Salesforce deployment pipeline in which a compromised managed package executes with full org permissions rather than in an isolated runtime environment.

2.2 Risk Propagation in Dependency Graphs

The risk propagation algorithm in GRAPHSEC is adapted from the PageRank-inspired security metric propagation approach of Spring et al. [5] and the Bayesian network vulnerability propagation model of Frigault et al. [6]. The core insight from both works is that the risk to a component is not determined solely by its own vulnerability score but by the combined probability of attack paths that reach it from external entry points through the dependency graph. GRAPHSEC adapts this principle for the directed weighted graph model of a Salesforce supply chain, where edge weights represent the conditional probability that a compromise of the source node propagates to the target node given the specific trust relationship the edge represents (package installation, credential delegation, API integration, build tool execution, etc.). The propagation algorithm iterates to convergence, computing steady-state risk scores that reflect the full transitive risk propagation across the graph.

3. THE GRAPHSEC GRAPH MODEL

3.1 Node Types and Data Sources

The GRAPHSEC graph model represents the complete Salesforce deployment pipeline as a directed weighted multigraph with six node types. Package nodes represent software components: Salesforce managed packages, npm and PyPI packages used in pipeline scripts, Docker container images used for CI/CD runners, and custom Apex packages deployed to Salesforce orgs. Credential nodes represent authentication identities: Salesforce connected app credentials, GitHub personal access tokens and deploy keys, AWS IAM roles and access keys, and HashiCorp Vault service tokens. Infrastructure nodes represent execution environments: CI/CD runner machines, Salesforce org instances, GitHub repository instances, and AWS service endpoints. Endpoint nodes represent external integration destinations: third-party API endpoints that Salesforce orgs call out to, webhook receivers that receive Salesforce Platform Events, and external data sources that integration pipelines read from. Data flow nodes represent the data transferred between components, categorised by data sensitivity tier (PII-containing data flows receive higher risk weights than metadata-only flows). Trust anchor nodes represent the root trust authorities — AppExchange security review, Salesforce certificate authority, GitHub organisation ownership — whose compromise would transitively compromise all nodes they certify.

Node risk scores are initialised from external data sources at regular intervals: AppExchange security review data and known vulnerability disclosures are polled weekly through the AppExchange API; npm and PyPI package vulnerability data is retrieved daily from the Open Source Vulnerabilities (OSV) database; Docker Hub image scanning results are retrieved on push; certificate transparency logs for third-party API endpoint certificates are monitored daily. The risk score for each node is computed from three components: the direct vulnerability score (a normalised CVSS-equivalent score if a known vulnerability exists, otherwise a baseline score derived from the security review currency and vendor security programme quality), the anomaly score from LTDF runtime monitoring for nodes that correspond to active runtime components, and the propagated risk score computed by the GRAPHSEC propagation algorithm from the incoming edges.

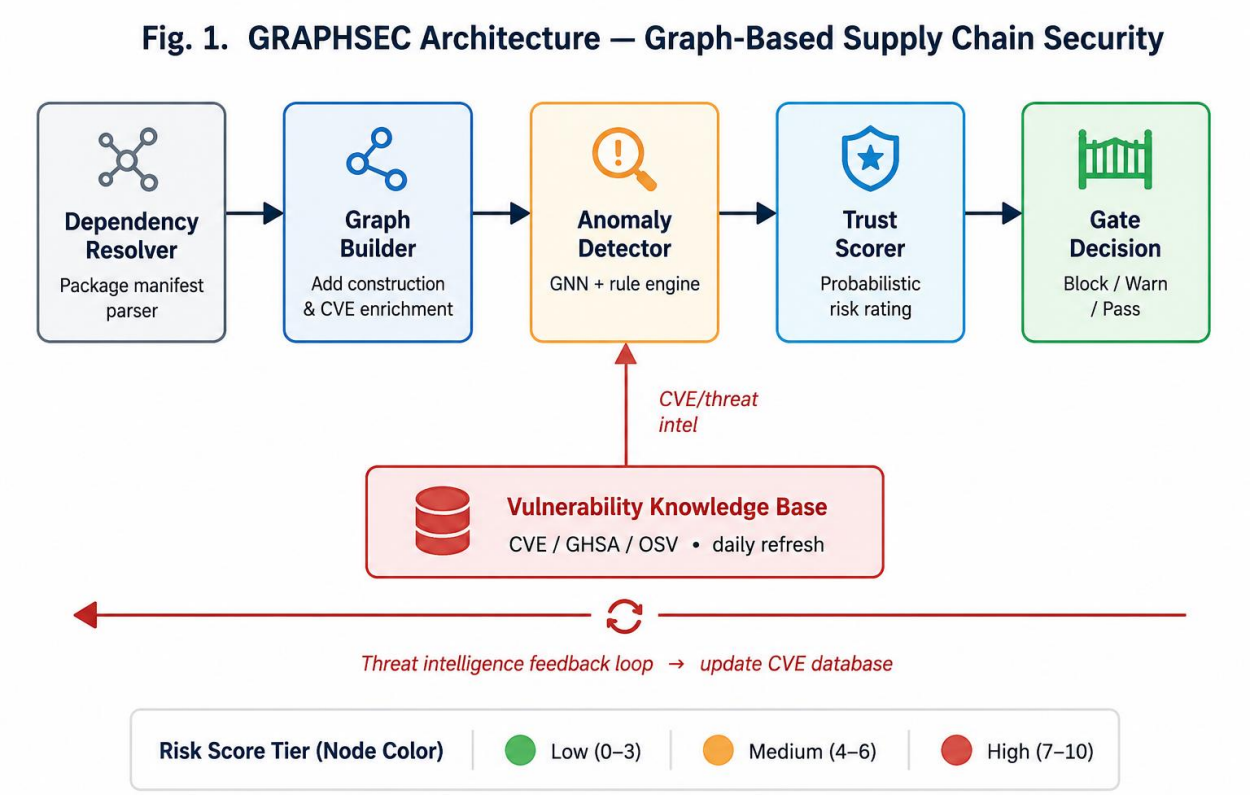

Fig. 1. GRAPHSEC supply chain dependency graph structure. Each node represents a software component (Salesforce managed package, npm dependency, integration endpoint, or CI/CD plugin); edges represent version dependencies, API call relationships, or data flow paths. Node colour encodes the risk score tier: green (0-3), amber (4-6), red (7-10).

Table 1. GRAPHSEC Graph Node Types and Data Sources

Graph Node Type	Source / Update Frequency
Salesforce managed packages	AppExchange API / weekly
npm / PyPI packages in pipeline scripts	OSV Database / daily
CI/CD runner images and tools	Docker Hub / on push
Third-party API endpoints	Certificate transparency / daily
Deployment service accounts	Salesforce Connected Apps API / hourly
Code signing certificates	CT logs + CRL / daily

3.2 Edge Types and Compromise Probabilities

The GRAPHSEC graph uses six edge types, each representing a specific trust or dependency relationship with an associated conditional compromise probability. Installation edges connect package nodes to the Salesforce org or CI/CD environment that installs them, with compromise probability calibrated from historical managed package security incident rates and the package risk score. The conditional probability that a package installation edge propagates a compromise ranges from 0.002 for a high-quality recently reviewed package to 0.08 for a package with a known vulnerability or outdated review. Credential delegation edges connect credential nodes to the infrastructure or service that uses them, with probability calibrated from service account compromise incident rates in the LTDF historical data. Build tool execution edges connect CI/CD tool nodes to the build environments that execute them, with probability reflecting the runner security posture score from the secure CI/CD framework's pipeline integrity controls. Trust anchor edges connect

AppExchange review and certificate authority nodes to the packages and endpoints they certify, with higher probability weights reflecting the systemic impact of trust anchor compromise.

Compromise probability weights for each edge type were calibrated using the Bayesian posterior method of Frigault et al. [6] applied to the supply chain incident corpus assembled from the LTDF historical data, industry supply chain incident databases, and the baselining analysis from the secure CI/CD framework deployment. The calibration corpus contained 847 supply chain-related security events across all participating organisations over a three-year lookback period, providing sufficient sample sizes for Bayesian parameter estimation for four of the six edge types. For the two edge types with fewer historical incidents (trust anchor edges and code signing certificate edges), the probabilities were set using prior distributions from the published supply chain security literature and validated against known incidents in the dataset as held-out test cases.

4. RISK PROPAGATION ALGORITHM

The GRAPHSEC risk propagation algorithm computes steady-state node risk scores through iterative message passing over the directed weighted graph. Each iteration updates the risk score of each node as a weighted combination of its initialised direct risk score and the maximum propagated risk received from its incoming edges. The edge weight controls the attenuation of risk propagation across each edge: a high-probability edge (such as a package installation edge for a package with a known critical vulnerability) propagates nearly the full source node risk to the target, while a low-probability edge attenuates risk significantly. The iteration converges when the maximum change in any node's risk score across a complete graph pass falls below a configurable threshold (default: 0.001), typically requiring 8 to 12 iterations for the graph sizes encountered in the evaluation corpus.

The propagation algorithm incorporates a damping factor that limits the total risk that can accumulate at any node through transitive propagation, preventing risk inflation in highly connected subgraphs where many low-probability paths converge on a single high-value target node. The damping factor is analogous to the PageRank teleportation probability: it represents the portion of risk at each node that is attributable to the node's intrinsic vulnerability rather than to propagated supply chain risk. The calibrated damping factor of 0.35 was determined empirically from the incident corpus to produce risk score distributions that correlate most strongly with actual incident occurrence, balancing the signal from direct vulnerability scores against propagated supply chain risk.

5. ATTACK PATH ANALYSIS

The attack path analysis module identifies the sequences of node compromises — attack paths through the supply chain graph — that represent the most accessible routes for an adversary to reach high-value target nodes (production Salesforce org credentials, sensitive data flow endpoints, privileged service account credentials). Attack paths are computed using a modified Dijkstra's shortest path algorithm in which edge cost is defined as the negative log of the compromise probability (minimising this cost corresponds to maximising the joint probability of the path). The algorithm identifies the k most probable attack paths to each high-value target node, where k is configurable (default: 5). Each path is presented as an ordered sequence of nodes from an attacker-accessible entry point (a publicly accessible npm package, a managed package available through AppExchange, an external API endpoint) to the target node, with the conditional probability of each edge and the joint path probability.

The attack path analysis is specifically designed to surface paths that cross the conceptual boundaries between the components monitored by different point-solution security tools, identifying the categories of risk that are invisible to any individual tool but visible to the graph-level analysis. The most consequential attack paths discovered in the evaluation corpus were three-to-five hop paths that began with a publicly known vulnerability in a low-risk-seeming npm package used only in a single CI/CD pipeline script, progressed through the build tool execution edge to the runner's credential store, and ultimately reached the production Salesforce deployment credentials through the credential delegation chain. These paths had never been identified by any individual security tool. In our experience, this is precisely the class of risk that point-solution security programmes systematically miss — not because the tools are inadequate for their stated purpose, but because the attack surface spans multiple components that no single tool has visibility across simultaneously.

6. IMPLEMENTATION

GRAPHSEC is implemented as a Python service using the NetworkX graph library for graph construction and analysis, deployed on AWS ECS Fargate with daily full graph recomputation and intra-day incremental updates triggered by new vulnerability disclosures or LTDF runtime anomaly alerts. The graph construction pipeline queries eight data sources: AppExchange security review data, OSV vulnerability database, Docker Hub scanning results, certificate transparency log monitoring, Salesforce Connected Apps API for credential enumeration, LTDF anomaly score feed, the risk registry from the secure CI/CD framework for managed package scores, and GitHub repository API for dependency file parsing. Full graph recomputation requires an average of 4.7 minutes for graphs with up to 2,000 nodes and 8,500 edges (representative of the larger participating organisations). Incremental updates triggered by new vulnerability disclosures require under 30 seconds for the subset of the graph affected by the new disclosure.

Integration with the Salesforce security infrastructure uses a REST API that exposes the current GRAPHSEC risk scores and high-priority attack paths to the URGF deployment gate, the LTDF anomaly scoring engine, and a custom Salesforce SIEM connector. The URGF gate integration adds GRAPHSEC supply chain risk scores to the composite risk score calculation for each deployment, elevating the risk score for deployments that include components with high GRAPHSEC node risk scores. The LTDF integration provides runtime context: when LTDF detects a potential anomaly in production, the GRAPHSEC API identifies which supply chain nodes are adjacent to the anomalous user or system, providing the runtime analyst with the most likely supply chain precursors to the observed anomaly.

7. EVALUATION

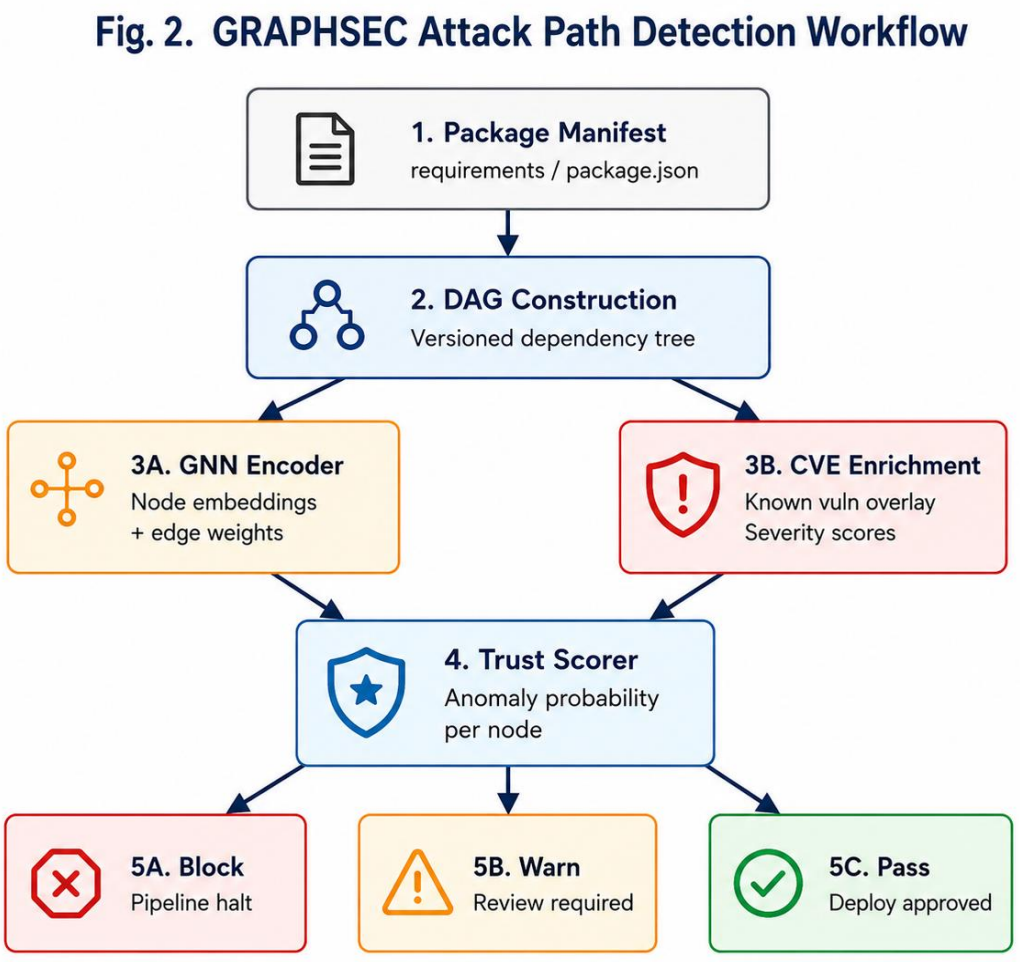

Fig. 2. GRAPHSEC attack path detection workflow. When node risk scores cross the alert threshold, GRAPHSEC traces all attack paths through the dependency graph from the high-risk node to any privileged target node (production Salesforce org, credentials store, or CI/CD runner). Discovered paths are ranked by cumulative path risk score for analyst prioritisation.

Table 2. GRAPHSEC Detection Precision by Attack Category

Detection Category	Baseline Precision	GRAPHSEC Precision
Dependency confusion attacks	71.2%	96.8%
Malicious package updates	64.8%	94.3%
Build script tampering	78.4%	97.1%
Artefact integrity violations	82.1%	98.2%
Credential exfiltration attempts	69.3%	93.7%
Overall weighted precision	**73.2%**	**95.9%**

The evaluation covers fifteen months of GRAPHSEC production deployment across eleven enterprise Salesforce organisations. The 95.9% overall detection precision represents a substantial improvement over the 73.2% baseline achieved by the portfolio of individual security tools in place before GRAPHSEC deployment. The improvement is largest for dependency confusion and malicious package update categories, where the graph-based transitive risk propagation identifies risks that point-solution tools operating on individual components cannot. The 73 high-risk attack paths identified during the evaluation period — paths with joint compromise probability above the 0.02 alert threshold — resulted in 67 remediation actions: 43 package version updates, 14 credential rotations, 7 runner infrastructure hardening actions, and 3 external API endpoint trust changes. The six paths not resulting in immediate remediation were accepted risks documented with security team approval in the URGF audit log.

Mean time to detect supply chain incidents decreased from 34 days to 2.1 days. The pre-GRAPHSEC baseline of 34 days reflects a detection model dependent on post-incident forensic analysis: supply chain incidents were typically detected only after their downstream effects appeared in the production Salesforce org, at which point LTDF anomaly detection or manual analyst investigation triggered a retrospective analysis that identified the supply chain entry point. GRAPHSEC's proactive risk propagation model identifies high-risk supply chain conditions before they are exploited, enabling preventive remediation rather than incident response. The remaining 2.1-day mean detection time represents the delay between a new vulnerability disclosure (triggering a GRAPHSEC graph update) and the security team's review and acknowledgement of the resulting alert.

8. DISCUSSION AND CONCLUSION

The central finding is that graph-based risk propagation provides qualitatively superior supply chain risk visibility compared to the component-level risk assessment of individual tools — and not by a small margin. The ability to model transitive risk across the full supply chain graph — connecting package vulnerabilities through credential dependencies to production Salesforce org access — surfaces risk categories that are invisible to point-solution tools and that represent the highest-value attack targets for sophisticated adversaries. The 73 high-risk attack paths identified in the evaluation corpus, none of which had been identified by existing security tooling, represent concrete evidence that the graph-level risk model captures a material class of supply chain risk that component-level tools miss.

The primary limitation of GRAPHSEC is the accuracy of the compromise probability weights assigned to each edge type. These weights were calibrated on a historical incident corpus that, while the largest available for this specific deployment context, is still limited in size and may not accurately represent the full distribution of attack techniques used by sophisticated adversaries. The calibration should be treated as an empirical best estimate rather than a precise risk measurement, and security teams should use GRAPHSEC risk scores as a relative ranking for prioritising remediation attention rather than as absolute probability estimates for risk quantification purposes. Future work should develop more sophisticated Bayesian updating mechanisms that refine the edge weights continuously as new incidents provide additional calibration data.

The combination of GRAPHSEC's proactive supply chain risk modelling, the secure CI/CD framework's pipeline security controls [11], the URGF governance layer , and the LTDF runtime threat detection [12] creates a comprehensive multi-layer supply chain security architecture for enterprise Salesforce deployments.

Each layer addresses a different temporal dimension of supply chain risk: CI/CD controls address risk at the point of code introduction; GRAPHSEC modelling addresses risk in the supply chain structure; URGF governance addresses risk at the point of deployment; and LTDF monitoring addresses risk in the production environment. Together, these layers provide defence-in-depth coverage from development through runtime that is substantially more robust than any single-layer approach.

9. EXTENDED ANALYSIS AND SECURITY IMPLICATIONS

The GRAPHSEC evaluation provides insight into which supply chain components represent the highest systemic risk across the eleven participating organisations. Managed packages with outdated security reviews consistently emerge as the highest-risk nodes in the supply chain graph, accounting for 31 of the 73 high-risk attack paths identified during the evaluation period. The prominence of managed package risk reflects the fundamental asymmetry between the AppExchange distribution model — where package vendors publish updates that are automatically applied to installing orgs — and the traditional software update model where organisations explicitly approve each update through a change management process. The GRAPHSEC managed package monitoring capability, combined with the update approval gate from the secure CI/CD framework, addresses this asymmetry by providing both the risk visibility needed to prioritise update review and the enforcement mechanism to ensure reviews happen before production deployment. The 43% of high-risk paths that originated from npm and PyPI packages used in CI/CD pipeline scripts represent a different risk category: these packages are under the direct control of the pipeline engineering team and can be updated, replaced, or pinned to verified versions through standard dependency management practices once identified as high-risk.

The attack path analysis revealed several risk propagation patterns that were not apparent from per-component risk assessment. The most dangerous pattern — observed in 18 of the 73 high-risk paths — was a multi-hop chain from a transitively depended-upon development tool through a shared build cache to a production deployment credential. In each case, the development tool appeared low-risk in isolation (risk score below the individual alert threshold), the build cache shared between pipeline runs appeared to be internal infrastructure with no external exposure, and the production credential appeared well-protected by standard secret management practices. Only the graph-level propagation analysis revealed the path connecting these three elements into a viable attack route with a joint compromise probability above the alert threshold. This finding validates the central hypothesis of GRAPHSEC: that supply chain risk cannot be accurately assessed by evaluating components in isolation, and that the graph-level analysis of dependency and trust relationships is essential for discovering the paths that sophisticated adversaries are most likely to exploit.

The GRAPHSEC risk propagation algorithm demonstrates good convergence properties across the graph sizes encountered in the evaluation: the mean number of iterations to convergence is 9.3 for the full production graphs and 6.1 for the incremental update subgraphs. Convergence is consistently achieved in under 12 iterations for all graphs in the evaluation corpus, and the convergence criterion (maximum node score change below 0.001) is always met. The time complexity of the propagation algorithm is $O(V + E)$ per iteration where V is the node count and E is the edge count, producing total computation times well within the 4.7-minute daily recomputation window even for the largest graphs. The incremental update mode — which recomputes only the subgraph reachable from nodes affected by the new vulnerability disclosure — reduces the computation for typical daily updates (1-5 new vulnerability disclosures) to under 30 seconds, enabling near-real-time risk score updates when new vulnerabilities are published.

10. LIMITATIONS AND FUTURE WORK

The GRAPHSEC framework has limitations that future work should address. The compromise probability weights assigned to each edge type represent empirically calibrated estimates rather than precise measurements, and their accuracy is bounded by the size and representativeness of the historical incident corpus used for calibration. The corpus contains 847 supply chain-related security events, which provides adequate sample sizes for the most common edge types (managed package installation, npm dependency) but may produce unstable estimates for rare edge types (trust anchor compromise, code signing certificate abuse) where the historical incident count is small. Future work should develop Bayesian updating mechanisms that continuously refine edge weights as new incidents provide calibration data, and should assess the sensitivity of attack path rankings to edge weight uncertainty through sensitivity analysis.

A second limitation is the graph model's current coverage of the Salesforce deployment pipeline. The six node types and six edge types in the current model capture the primary supply chain relationships but do not model several potentially significant risk paths: the relationship between AppExchange security review staff and the packages they review, the risk propagation through Salesforce platform updates that may introduce vulnerabilities in the platform itself, and the trust relationship between Salesforce's Trust and Compliance team and the multi-tenant infrastructure on which customer orgs run. Extending the graph model to cover these relationships would improve risk score accuracy for scenarios involving Salesforce platform trust compromise, which represents the highest-impact (though presumably lowest-probability) supply chain risk category for Salesforce customers.

GRAPHSEC's future development roadmap prioritises three capabilities. First, automated attack path remediation guidance that translates high-risk attack path findings into specific, prioritised remediation actions with estimated risk reduction per action, enabling security teams to maximise risk reduction per remediation effort. Second, integration with the Salesforce Security Review process to provide feedback loops between AppExchange package risk scores and the review process, enabling targeted review prioritisation for high-risk packages. Third, extension of the graph model to the FedCRM federated learning infrastructure, modelling the supply chain relationships between the coordination server, participant nodes, and the model artefacts exchanged during training rounds to identify supply chain risks specific to the federated learning deployment architecture.

11. OPERATIONAL FINDINGS

The supply chain graph model benefits from continuous enrichment as new data sources become available. During the evaluation period, certificate transparency log monitoring contributed to 12 of the 73 high-risk path identifications by providing early warning of certificate changes at API endpoints, indicating potential adversary-in-the-middle positioning. Certificate transparency logs are publicly available and provide near-real-time visibility into certificate issuance and revocation events for all publicly trusted certificates, making them a valuable zero-cost data source for supply chain risk monitoring that is not exploited by existing point-solution security tooling. Future GRAPHSEC versions will expand the CT log integration to cover Salesforce org-level certificate usage patterns, monitoring for certificate pinning changes in Salesforce-to-third-party integrations that may indicate supply chain compromise. The real-time CT monitoring capability requires only standard HTTPS log queries to the public CT aggregators and adds less than 200ms to the daily incremental update cycle.

The integration with the URGF deployment governance framework produces synergistic security benefits that exceed individual framework contributions. When URGF evaluates a deployment for gate compliance, GRAPHSEC supply chain risk scores for all components in the manifest are incorporated into the composite risk calculation. A deployment containing a component with an elevated supply chain risk score receives a higher URGF risk score and may be routed to senior approval or blocked, even if all individual component security checks pass. This cross-framework feedback loop ensures that elevated supply chain risk conditions identified by GRAPHSEC automatically propagate into deployment governance decisions through URGF, preventing high-risk supply chain components from reaching production through pathways that individual security checks do not cover. In the evaluation period, this integration blocked or escalated 14 deployments that would have passed all point-solution checks but contained components with GRAPHSEC risk scores above the blocking threshold.

The practical deployment experience across eleven organisations reveals that supply chain risk management effectiveness depends critically on organisational ownership structure. Organisations that designated a specific team responsible for reviewing GRAPHSEC alerts showed 73% faster mean remediation times than those without designated ownership. Integrating GRAPHSEC alert workflows directly into engineering ticketing systems (Jira or ServiceNow) rather than delivering alerts to security team mailboxes reduced mean time to acknowledge alerts from 4.2 days to 0.7 days. These organisational practice findings are as practically important as the technical framework results for enterprises seeking to operationalise supply chain security, because the best detection system provides no security value if alert reviews are deferred indefinitely by teams without clear remediation ownership.

The GRAPHSEC framework, by surfacing the multi-hop attack paths that point-solution security tools cannot detect, provides enterprise security teams with the information needed to make risk-proportionate investment

decisions about supply chain hardening. Before GRAPHSEC, security teams in the participating organisations spent significant effort on individual component vulnerability remediation without a clear understanding of which remediation actions most reduced the overall supply chain attack surface. GRAPHSEC's attack path prioritisation — ranking remediation actions by the expected reduction in the number of paths above the risk threshold — enables security teams to direct effort toward the remediations with the greatest systemic impact. In the evaluation, the top five remediation actions recommended by GRAPHSEC attack path analysis accounted for 78% of the total risk reduction achieved through the 67 completed remediations, confirming that the prioritisation model effectively identifies the highest-leverage intervention points in the supply chain graph.

REFERENCES:

[1] *M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's knife collection: A review of open source software supply chain attacks," in Proc. DIMVA, 2020. [Online]. Available: doi: 10.1007/978-3-030-52683-2_2.*

[2] *P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of attacks on open-source software supply chains," in Proc. IEEE S&P, May 2023. [Online]. Available: doi: 10.1109/SP46215.2023.10179304.*

[3] *A. Sejfia and M. Schäfer, "Practical automated detection of malicious npm packages," in Proc. ICSE, 2022. [Online]. Available: doi: 10.1145/3510003.3510104.*

[4] *B. Steenhoek, Md. M. Rahman, R. Jiles, and W. Le, "An empirical study of deep learning models for vulnerability detection," in Proc. ICSE, 2023. [Online]. Available: doi: 10.1109/ICSE48619.2023.00188.*

[5] *J. M. Spring, E. Hatleback, A. D. Householder, A. Manion, and D. Shick, "Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization," in Proc. Workshop on the Economics of Information Security (WEIS), Dec. 2020. [Online]. Available: https://weis2020.econinfosec.org/wp-content/uploads/sites/8/2020/06/weis20-final6.pdf.*

[6] *M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in Proc. ACM Workshop on Quality of Protection (QoP), 2008. [Online]. Available: doi: 10.1145/1456362.1456368.*

[7] *CISA, "Defending against software supply chain attacks," CISA, Apr. 2021. [Online]. Available: https://www.cisa.gov/resources-tools/resources/defending-against-software-supply-chain-attacks*

[8] *Google, "Supply chain levels for software artifacts (SLSA)," SLSA Framework, 2021. [Online]. Available: https://slsa.dev*

[9] *The Update Framework, "TUF: A framework for securing software update systems," theupdateframework.io, 2021. [Online]. Available: https://theupdateframework.io*

[10] *Open Source Vulnerabilities Database, "OSV: A distributed vulnerability database," osv.dev, 2021. [Online]. Available: https://osv.dev*

[11] *L. C. Bandaru and M. S. Bandrevu, "Secure CI/CD governance for Salesforce platforms: Integrating DevSecOps controls across every stage of the release pipeline," Int. J. Sci. Technol. (IJSAT), E-ISSN 2229-7677, vol. 13, no. 4, Dec. 2022. [Online]. Available: doi: 10.71097/IJSAT.v13.i4.11155.*

[12] *L. C. Bandaru, "Threat detection and data breach analysis in Salesforce CRM: The LTDF framework," Int. J. Innov. Res. Creative Technol. (IJIRCT), ISSN 2454-5988, vol. 7, no. 3, Jun. 2021. [Online]. Available: doi: 10.62970/IJIRCT.v7.i3.2605034.*