International Journal of Engineering Research and Modern Education (IJERME)
International Peer Reviewed - Refereed Research Journal, Website: www.rdmodernresearch.com
Impact Factor: 7.018, ISSN (Online): 2455 - 4200, Volume 9, Issue 2, July - December, 2024

ASSESSING ASSOCIATIONS BETWEEN ATTACK MECHANISMS AND DATA EXFILTRATION IN SOFTWARE SUPPLY CHAINS ACROSS GLOBAL ECOSYSTEMS

Divya Vishwanath* & Paul Johnson Asamoah**
* Janardhan Rai Nagar Rajasthan Vidyapeeth, Udaipur, Rajasthan, India
** Power Life University College, Accra, Ghana, West Africa

Cite This Article: Divya Vishwanath & Paul Johnson Asamoah, "Assessing Associations Between Attack Mechanisms and Data Exfiltration in Software Supply Chains Across Global Ecosystems", International Journal of Engineering Research and Modern Education, Volume 9, Issue 2, July - December, Page Number 79-87, 2024.

Copy Right: © R&D Modern Research Publication, 2024 (All Rights Reserved). This is an Open Access Article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract:

This analysis examines how documented data exfiltration differs across compromise mechanisms in global software supply chains. We analyze 175 incident records from the Atlantic Council database covering 2010 to 2023 through descriptive statistics, cross tabulations and an exploratory reporting completeness restriction. Documented exfiltration appears in 78.9 percent of incidents. Its proportion is 17.56 percentage points lower among incidents with coded account access and 6.93 points lower among those involving self-signed or unsigned software than among their respective comparison groups. Incidents involving stolen or purchased certificates show a proportion 1.33 points higher. Excluding nine records with unknown or other impact information preserves the first two directions but increases the certificate contrast to 9.16 points. These patterns distinguish opportunities to cross trust boundaries from documented downstream consequences without establishing causal pathways. The contribution is an outcome specific comparison showing that mechanism frequency, exfiltration proportions and reporting completeness capture different dimensions of incident evidence. Security teams and reporting bodies should preserve these distinctions when assessing exposure, comparing incidents and allocating investigative resources.

Key Words: Cyber Security; Data Exfiltration; Descriptive Analysis; Software Supply Chains; Trust Boundaries.

1. Introduction:

Software supply chain attacks exploit the relationships that allow organizations to acquire, update and reuse software. A compromised upstream account or artifact can expose downstream users who did not participate in the original security failure. Yet identifying how an attack entered a supply chain does not establish what information it extracted. This distinction matters for incident assessment: an entry mechanism describes an opportunity for compromise, whereas documented exfiltration records a specific consequence. Treating these categories as equivalent can obscure differences between attacks and encourage security priorities based on mechanism labels rather than observed impacts. Research on software ecosystems explains why upstream weaknesses can have consequences beyond individual projects. Decan et al., 2019, identify concentrated reverse dependencies and increasing fragility across seven packaging ecosystems. Zerouali et al., 2022, show how vulnerable packages expose direct and indirect dependents within npm and RubyGems. These findings establish the importance of interdependence for understanding exposure. However, dependency exposure is not equivalent to malicious execution or information theft. The distinction becomes operationally important when organizations must determine whether a reported supply chain incident warrants investigation of sensitive information access, software integrity or other consequences.

The closest research also distinguishes security indicators from the behavior they are intended to identify. Ponta et al., 2020, assess vulnerable dependencies through code reachability within applications, demonstrating why metadata alone can provide an incomplete account of practical exposure. Andreoli et al., 2023, examine characteristic behaviors of software supply chain attacks and develop an investigative framework for identifying malicious activity. These approaches clarify vulnerability applicability and malicious behavior, respectively. The reviewed evidence does not establish how documented exfiltration proportions differ across account access, self-signed or unsigned software, and stolen or purchased certificates within a common incident collection. That comparison addresses a narrower question than attack detection: whether different recorded compromise mechanisms exhibit different profiles of documented information loss. A trust boundary perspective provides the conceptual basis for this comparison. Account access can allow attackers to modify software through permissions associated with an established account. Self-signed or unsigned software concerns acceptance of an artifact without independent publisher assurance. Stolen or purchased certificates concern appropriation of a recognized signing identity. These mechanisms differ in the authority they exploit, but none determines the payload objective. Cooper et al., 2018, explain that code signing provides integrity and signer authentication. Those functions establish why signing authority matters without implying that signed software cannot contain malicious functionality. The theoretical expectation is therefore that entry mechanisms and exfiltration outcomes require separate measurement, while the direction of their association remains an empirical question.

This analysis aims to characterize the relationship between these three recorded attack mechanisms and documented data exfiltration across global software ecosystems. The central research question asks how the proportion of incidents with documented exfiltration differs between records with and without each mechanism coded. Specifically, we compare account access, self-signed or unsigned software, and stolen or purchased certificates against the same binary outcome. We also examine whether excluding incidents with unknown or other impact information changes the direction or magnitude of these comparisons. This approach distinguishes variation in recorded outcomes from claims about causal effects or the effectiveness of security controls. The empirical setting comprises 175 incident records classified as attacks in version 3 of the Atlantic Council software supply chain security database. Retained incident dates extend from July 2010 to June 2023. Loomis et al., 2023, provide the underlying collection of reported attacks and disclosures, including attack vectors, distribution channels, impacts and supporting references. We preserve the source classifications and use descriptive statistics and cross tabulations to compare conditional

exfiltration proportions. Each indicator identifies whether a feature was coded in the source; an uncoded feature does not establish confirmed absence. The collection supports comparisons among recorded incidents rather than estimates of attack prevalence across all software users or ecosystems.

The findings contribute an outcome specific distinction between mechanism prevalence, documented exfiltration and reporting completeness. Account access exhibits the largest negative exfiltration contrast, while self-signed or unsigned software exhibits a smaller negative contrast. Certificate acquisition shows little separation in the full sample, but its contrast increases when records with incomplete impact information are excluded. Thus, the most frequent selected mechanism does not produce the largest outcome contrast, and the certificate comparison depends more strongly on sample composition. These findings qualify interpretations that assign a uniform information theft profile to broad attack categories. Their practical contribution is to support incident records and assessments that distinguish the compromise mechanism, documented consequences and completeness of impact evidence. Their international relevance rests on software relationships that cross organizational and geographical boundaries, while the observed proportions remain specific to the accessible incident collection. The remainder of the article develops the conceptual framework and research questions in Section 2, presents the data in Section 3, and explains the methods in Section 4. Section 5 reports the findings, Section 6 discusses their significance, and Section 7 presents the conclusions and implications.

2. Theory, Conceptual Framework and Research Questions Development:

This analysis applies a trust boundary perspective to software supply chain attacks across global ecosystems from 2010 to 2023. Developers, repositories, software vendors and users depend on decisions made outside their direct control. These relationships allow software to circulate through established distribution channels, but they also create opportunities for attackers to exploit upstream permissions and downstream acceptance. Hammi et al., 2023, identify the expansion of digital supply chain connections as a source of security exposure across interconnected technologies. This provides a system level explanation for examining how different attack mechanisms relate to documented data exfiltration. The framework distinguishes the opportunity to introduce malicious software from the outcome recorded after an attack. Account access, self-signed or unsigned software, and stolen or purchased certificates represent three observed mechanisms through which attackers can cross trust boundaries. Documented data exfiltration represents the recorded transfer of victim information to an attacker. Boyens et al., 2022, explain that limited visibility into how technology is developed, integrated and deployed creates supply chain risk. Applied here, this perspective suggests that upstream trust failures can expose downstream information, while the resulting impact depends on what the malicious software is designed and permitted to do.

We operationalize these relationships using 175 recorded attacks. Each mechanism is a separate binary indicator rather than a component of an assumed latent scale. The outcome equals one when the source codes data extraction and zero when it does not. Zero therefore denotes an uncoded outcome rather than confirmed absence. The mechanisms can coexist within an incident, so the framework treats them as related but distinct explanatory variables. The research questions concern associations within reported attacks. Account access captures incidents in which attackers obtained access to an account with permission to modify affected software. Its theoretical relevance lies in the distinction between possessing credentials and exercising legitimate authority. An attacker who acquires an authorized account may alter software through an accepted workflow, allowing malicious content to inherit the apparent legitimacy of its publisher. Rose et al., 2020, establish that authentication and authorization require separate decisions and that account ownership does not justify implicit trust. This principle explains why compromised access can provide a route from upstream modification to downstream exposure. However, access to a publishing account does not determine the payload objective. The same permission can support information theft, unauthorized execution or other recorded impacts. Account compromise therefore provides a plausible enabling mechanism for exfiltration without establishing that exfiltration should occur more frequently than under other attack mechanisms. We examine the association without imposing a directional expectation.

RQ1: How does the proportion of attacks with documented data exfiltration differ between incidents with and without coded account access?

Self-signed or unsigned software captures a different trust boundary. Rather than relying on control of an established account, this mechanism concerns software accepted without a signature from an independently trusted publisher. The theoretical pathway involves the admission of an artifact whose origin has limited assurance. Souppaya et al., 2022, identify protection against unauthorized access and tampering as central requirements of secure software development. Their framework supports distinguishing the conditions under which malicious software enters a workflow from the actions it performs after execution. Limited publisher assurance can permit software containing information collection functions to reach users. Yet it can also characterize attacks designed for advertising, mining or disruption. Conversely, software carrying a legitimate signature can still contain malicious functionality. The signature category therefore identifies an entry mechanism rather than a fixed payload purpose. Its relationship with documented exfiltration remains an empirical question.

RQ2: How does the proportion of attacks with documented data exfiltration differ between incidents with and without coded self-signed or unsigned software?

Stolen or purchased certificates concern the appropriation of an external assurance mechanism. A certificate can make malicious software appear connected to an accepted publisher even when the attacker lacks legitimate authority. Cooper et al., 2018, explain that code signing provides evidence of integrity and signer identity. These functions establish why certificate misuse matters, but they do not establish that signed software is harmless. The theoretical mechanism is the acceptance of an apparently authenticated artifact whose signing authority has been abused. Certificate misuse may help an attacker deliver an information stealing payload through a trusted channel. However, the same assurance can support payloads with other objectives, and obtaining a certificate does not itself establish access to victim information. A directional claim would therefore exceed the mechanism established by the available evidence. We retain a separate question for this association.

RQ3: How does the proportion of attacks with documented data exfiltration differ between incidents with and without coded stolen or purchased certificates?

The conceptual framework connects three observed attack mechanisms to one observed outcome through a shared trust boundary explanation. Account access concerns control over modification permissions, self-signed or unsigned software concerns limited publisher assurance, and certificate misuse concerns the appropriation of signing authority. We examine each mechanism separately and jointly in relation to documented data exfiltration. These relationships represent statistical associations among recorded incidents. They allow positive, negative, null or inconclusive outcomes without assuming that an entry mechanism determines the purpose of an attack.

3. Data:

This analysis documents the source records, sample selection and measurement of attack mechanisms and documented data exfiltration. It uses incident level observations to describe how recorded exfiltration varies across three mechanisms within global software supply chains.

3.1 Data Sources, Population, and Sample Construction:

The empirical source is the Atlantic Council software supply chain security database, version 3, released in September 2023. The publicly downloadable workbook contains incident descriptions, dates, attack classifications, affected software, distribution channels, impact indicators and supporting reference links. Loomis et al., 2023, describe the database as a collection of reported software supply chain attacks and vulnerability disclosures. We use its Data worksheet as the sampling frame and retain its original row numbers for traceability. The accompanying provenance worksheet records the extraction date separately from the observation period. The downloaded materials do not specify a dataset licence. The target population comprises software supply chain attacks across global software ecosystems during 2010 to 2023. The accessible frame consists of 250 named records in the released workbook. This frame is a curated incident collection rather than a probability sample or a census of all attacks. Coverage spans proprietary applications, open source software, firmware and attacker supplied applications. Geographical coverage follows the reported incidents rather than a country sampling design.

The unit of analysis is one source incident record. The data form an incident cross section with historical dates, rather than an annual panel. Retained dates extend from 15 July 2010 to 15 June 2023, with observations represented in every calendar year within that interval. Dates identify estimated attack commencement or discovery when commencement is unavailable. They do not establish a uniform observation frequency or a complete annual reporting window. Selection retained records explicitly classified as Attack, with a nonempty incident name and a numeric source year between 2010 and 2023. These rules were implemented during dataset construction, before calculating the descriptive comparisons reported here. No subsequent eligibility changes were introduced.

Table 1: Sample construction

Stage	Incident records	Observations	Reason or Rule
Accessible source frame	250	250	Named entries in the version 3 Data worksheet
Disclosure records excluded	72	72	Source classification equals Disclosure
Mixed classifications excluded	3	3	Source classification equals Attack/Disclosure
Records excluded by year restriction	0	0	Source year falls outside the specified interval
Eligible attack records	175	175	Source classification equals Attack
Additional losses during cleaning	0	0	Selected analytical fields contain no missing values
Final descriptive sample	175	175	Complete records for all three mechanisms and the outcome

The source webpage reports 168 attacks and 82 disclosures, whereas the released workbook contains 175 Attack, 72 Disclosure and three Attack/Disclosure classifications. Selection follows the classifications in the downloaded records. These counts describe incident entries, not affected organizations, software packages, installations or victims.

3.2 Data Integration, Cleaning, and Quality Assurance:

We extracted the analytical variables from a single source worksheet. Each analytical record retains its original source row number, incident name, date, year, impact tags and reference URLs. Source row numbers provide a unique connection between the analytical worksheet and the retained source records. No external dataset merge, temporal aggregation or frequency conversion was performed. Verification confirmed 175 unique source row numbers and 175 unique analytical identifiers. Incident names contain 174 distinct values. Two records share the name Malicious Chrome Extensions but have different dates and reference URLs. Both were retained because matching names alone do not establish duplicate incidents. No complete analytical rows were duplicated.

All three mechanism indicators, the exfiltration indicator and the unknown impact flag match their corresponding source fields. Each contains only zero or one. There are no missing values in these five fields across retained incidents or represented years. No imputation, interpolation, winsorization or replacement of missing observations with zero was applied. The distinction between an uncoded feature and a confirmed absence remains central. A zero preserves the source classification and does not establish that an attack lacked the mechanism or outcome. Nine incidents carry an unknown or other impact flag. Four also have documented data extraction, demonstrating that incomplete impact information can coexist with a recorded outcome.

The analytical workbook preserves source values alongside their definitions and identifiers. This supports inspection of the transformation from source records to analytical fields, consistent with the emphasis on provenance and contextual metadata in Goodman et al., 2014. The checks establish extraction consistency rather than independent verification of every underlying incident report. The database provides a retrospective account of incidents. We do not reconstruct what information was available at each incident date. Reporting, discovery and source inclusion can therefore influence sample composition. Annual record counts describe the composition of the collection rather than changes in the underlying incidence of attacks.

3.3 Variable Measurement, Construction, and Validation:

The explanatory variables are three observed attack mechanisms: account access, self-signed or unsigned software, and stolen or purchased certificates. The outcome is documented data exfiltration, measured through the source Data Extraction

indicator. The Atlantic Council, 2020, codebook defines these categories as account access with editing permissions, attacker signed software or software requiring no signature, acquisition of a legitimate signing certificate, and extraction of victim information to an attacker-controlled server. The indicators remain separate because they describe distinct mechanisms and can coexist within an incident. We construct no composite index or latent scale. A value of one indicates source coding of the specified feature. Higher values therefore represent recorded presence, without implying greater attack intensity.

Table 2: Variable dictionary

Variable or Indicator	Analytical Role	Definition and Construction	Source and Identifier	Unit and Frequency	Transformation and Direction	Missing Value Rule	Coverage and Limitation
Documented data exfiltration	Outcome	Source codes extraction of victim information	Atlantic Council version 3, Data Extraction	Binary indicator per incident	Original coding retained; one denotes documented extraction	No missing values; zero retained as not coded	175 incidents; does not measure information volume
Account access	Explanatory variable	Attacker accessed an account with software editing permissions	Atlantic Council version 3, Account Access	Binary indicator per incident	Original coding retained; one denotes coded account access	No missing values	175 incidents; access privileges are not quantified
Self signed or unsigned software	Explanatory variable	Attacker signed software or no signature was required	Atlantic Council version 3, Self signed/Unsigned	Binary indicator per incident	Original coding retained; one denotes the source category	No missing values	175 incidents; combines two signing conditions
Stolen or purchased certificates	Explanatory variable	Attacker acquired a legitimate code signing certificate	Atlantic Council version 3, Stolen/Purchased Certificate	Binary indicator per incident	Original coding retained; one denotes certificate acquisition	No missing values	175 incidents; theft and purchase are not separated
Unknown or other impact	Quality indicator	Source records unknown or other impact information	Atlantic Council version 3, Unknown or Other	Binary indicator per incident	Original coding retained	No missing values	175 incidents; can coexist with documented extraction
Source year	Eligibility and coverage field	Original numeric incident year	Atlantic Council version 3, Year	Calendar year per incident	Restricted to 2010 through 2023	No missing values among retained records	Historical classification rather than annual follows up
Source row number	Provenance identifier	Original row position in the Data worksheet	Atlantic Council version 3, Data worksheet	Identifier per incident	Preserved without analytical scaling	No missing values	Unique within the downloaded release

The certificate measure identifies misuse of signing authority rather than the security quality of signed software. Cooper et al., 2018, distinguish the integrity and signer authentication functions of code signing. This distinction supports interpreting certificate acquisition as an attack mechanism without treating a signature as evidence that software is harmless.

Table 3: Descriptive statistics

Indicator	Valid Observations	Records Coded One	Mean	Sample Standard Deviation	Median	Minimum	Maximum
Documented data exfiltration	175	138	0.789	0.409	1	0	1
Account access	175	34	0.194	0.397	0	0	1
Self signed or unsigned software	175	85	0.486	0.501	0	0	1
Stolen or purchased certificates	175	25	0.143	0.351	0	0	1
Unknown or other impact	175	9	0.051	0.222	0	0	1

For these binary indicators, the mean equals the proportion coded one. Documented exfiltration appears in 78.9 percent of retained records. Account access appears in 19.4 percent, self-signed or unsigned software in 48.6 percent, and stolen or purchased certificates in 14.3 percent. These proportions describe source coding within the selected collection.

3.4 Analytical Framework, Identification, and Reproducibility:

The implemented analysis is descriptive. It calculates valid counts, proportions, means, sample standard deviations, medians and ranges, followed by cross tabulations of each mechanism against documented exfiltration. The comparison of interest is the difference in recorded exfiltration proportions between incidents coded one and zero for each mechanism. Both groups come from the same 175 incident sample. Documented exfiltration appears in 22 of 34 account access records, or 64.7 percent, compared with 116 of 141 records without coded account access, or 82.3 percent. The corresponding proportions are 75.3 percent and 82.2 percent for self-signed or unsigned software, and 80.0 percent and 78.7 percent for stolen or purchased certificates. These comparisons establish observed patterns without attributing the differences to the mechanisms themselves.

Interpretation remains conditional on inclusion in the incident collection and the completeness of source reporting. Mechanisms can overlap, incidents can share campaigns or infrastructure, and reporting practices can differ across software ecosystems. The descriptive contrasts therefore do not isolate causal effects or establish population probabilities. Kass et al., 2016, emphasize matching statistical claims to the question, data and assumptions; here, the reported evidence consists of sample proportions rather than causal estimates. An exploratory completeness check excluded all nine records carrying the unknown or other impact flag. This retained 166 incidents, including 134 with documented exfiltration, giving a proportion of 80.7 percent compared with 78.9 percent in the full sample. The check assesses the aggregate proportion under a stricter reporting criterion. It does not establish that the remaining records contain complete impact information. Reproducibility rests on the retained source rows, unique identifiers, variable dictionary, eligibility rules and incident reference links. The same incident unit, binary coding

and coverage restrictions apply to every analytical variable. These choices make the descriptive comparisons auditable while preserving the distinction between recorded evidence and unobserved attack characteristics.

4. Methods:

4.1 Research Design and Analytical Scope:

This analysis uses a retrospective descriptive design to examine documented data exfiltration across three recorded software supply chain attack mechanisms. The unit of analysis is an incident. The analytical target is the difference in recorded exfiltration proportions between incidents with and without each mechanism coded. Comparisons use variation across incidents rather than repeated observations of organizations, software packages or countries. The sample construction in Table 1 supplies 175 eligible incidents covering 2010 through 2023. The variable dictionary in Table 2 defines the outcome and mechanism indicators. Each comparison uses the same incident collection, allowing differences across mechanisms to be interpreted against a consistent sample boundary. The design answers the research questions through observed proportions. It does not identify the independent causal contribution of a mechanism because attack characteristics can overlap and source inclusion is not random. Differences can also reflect payload objectives, affect software or report completeness. Accordingly, the analytical choices prioritize alignment between the research question and the available observations, consistent with Kass et al., 2016.

4.2 Descriptive Specification and Measurement Alignment:

We calculate frequencies, proportions, means, sample standard deviations, medians and observed ranges for the binary indicators. Standard deviations describe dispersion among incident records rather than uncertainty around population estimates. Each incident receives equal weight, irrespective of the number of affected installations, organizations or victims. For each mechanism, we construct a two-by-two cross tabulation against documented exfiltration. Equation 1 defines the conditional proportion used in these comparisons.

$$\hat{p}_{jx} = \frac{\sum_{i=1}^{N} Y_i \; 1_{X_{ij}=x}}{\sum_{i=1}^{N} 1_{X_{ij}=x}}$$

Here, $\hat{p}_{jx}$ is the recorded exfiltration proportion for mechanism jwithin coding group x. The index i identifies an incident, Nis the analytical sample size, $Y_i$equals one when exfiltration is coded, and X_{ij}records mechanism j. The index jidentifies account access, self signed or unsigned software, or stolen or purchased certificates. The group value \(x\) is zero or one. The indicator $1_{X_{ij}=x}$equals one when the incident belongs to the specified group and zero otherwise. Equation 2 expresses the comparison in percentage points.

$$\widehat{\Delta}_j = 100\left[\hat{p}_{j1} - \hat{p}_{j0}\right]$$

Here, $\widehat{\Delta}_j$ is the percentage point difference between incidents with mechanism \(j\) coded and incidents without that coding. Positive values indicate a higher documented exfiltration proportion in the coded group; negative values indicate a lower proportion. This contrast is not a percentage change. Both equations preserve the source distinction between coded presence and an uncoded feature. They do not reinterpret zero as independently confirmed absence. Mechanisms remain separate observed indicators rather than components of a composite index. Their overlapping membership means that the contrasts cannot be added or interpreted as independent shares of the outcome.

4.3 Processing, Verification and Computational Procedures:

Processing began with the analytical worksheet and its corresponding retained source records. We checked record counts, source row identifiers, incident names, date coverage and the permitted values of each analytical indicator. We then matched the mechanism, outcome and impact completeness fields to their original source columns using source row numbers. Duplicate checks distinguished repeated names from repeated records. Two entries sharing an incident name were retained because their dates and reference URLs differed. Verification confirmed unique analytical identifiers and source row identifiers. All selected indicators contained valid binary values and matched the retained source fields. The analytical sequence comprised descriptive summaries, mechanism specific cross tabulations, conditional proportions under Equation 1 and percentage point differences under Equation 2. We also counted how many selected mechanisms were coded within each incident to establish whether the comparison groups overlapped. No missing values required imputation or interpolation, and no observations were removed through outlier treatment.

Calculations used Python version 3.12.14. The openpyxl package, version 3.1.5, read stored workbook values, while the standard statistics and collections modules generated summaries and frequency counts. Sample standard deviations used the sample size less one as the denominator. Calculations retained full precision until presentation. Percentages and contrasts were rounded to two decimal places in the comparison tables, while means and standard deviations were rounded to three decimal places. The workbook preserves analytical records, corresponding source records, source identifiers and measurement definitions. This structure supports inspection of the extraction and classification steps, reflecting the importance of linking data with provenance and processing information emphasized by Goodman et al., 2014.

4.4 Sensitivity Analysis and Interpretive Limits:

An exploratory sensitivity analysis examined the influence of incomplete impact information. We excluded every incident carrying the source unknown or other impact flag, including flagged incidents that also contained documented exfiltration. The restriction retained 166 incidents. We then recalculated the overall exfiltration proportion and all three mechanism comparisons using Equations 1 and 2. The comparison between the full and restricted samples evaluates changes in group composition, direction and magnitude. For each mechanism, we inspected the number of removed incidents and their outcome coding. This distinguishes a change produced by selective record removal from one produced by a different analytical formula. The restricted sample is an alternative reporting completeness criterion rather than a verified set of fully observed incidents.

Results are reported as descriptive sample contrasts without significance thresholds. The magnitude of a contrast does not establish its precision beyond the collection, and a small contrast does not demonstrate equivalence. Greenland et al., 2016,

emphasize that statistical interpretation depends on assumptions and analytical decisions; here, interpretation remains bounded by retrospective source coding and incident selection. Together, the common sample, unchanged binary definitions, source matching and explicit completeness restriction provide an auditable answer to the research questions. The method distinguishes documented outcome patterns from causal explanations and shows how reporting composition affects those patterns.

5. Findings:

This analysis examines how documented data exfiltration varies across account access, self-signed or unsigned software, and stolen or purchased certificates. The findings address the three research questions through incident counts, conditional proportions and differences in percentage points. An exploratory sensitivity analysis assesses whether incomplete impact information changes these comparisons.

5.1 Sample Characteristics and Measurement:

The analytical sample contains 175 incident records dated between July 2010 and June 2023. Selection retained records explicitly classified as attacks and excluded 72 disclosures and three mixed attack and disclosure records. All retained incidents have complete values for the three mechanism indicators and documented data exfiltration. The outcome records whether the source identifies data extraction. Each mechanism indicator records whether the corresponding attack feature appears in the source coding. A zero denotes that the feature was not coded, rather than independently confirmed absence. Consequently, the comparisons concern documented characteristics within the incident collection. Documented data exfiltration appears in 138 incidents, representing 78.9 percent of the sample. Self signed or unsigned software is the most frequent selected mechanism, appearing in 85 incidents, followed by account access in 34 and stolen or purchased certificates in 25.

Table 4: Distribution of documented attack characteristics

Indicator	Valid Observations	Records Coded One	Percentage Coded One	Mean	Sample Standard Deviation
Documented data exfiltration	175	138	78.9	0.789	0.409
Account access	175	34	19.4	0.194	0.397
Self signed or unsigned software	175	85	48.6	0.486	0.501
Stolen or purchased certificates	175	25	14.3	0.143	0.351
Unknown or other impact	175	9	5.1	0.051	0.222

The mechanisms are not mutually exclusive. Ten incidents contain two selected mechanisms, while 124 contain one and 41 contain none. None contains all three. These overlaps mean that separate comparisons cannot identify the independent contribution of each mechanism. The distinction between the research question and the evidence available for answering it follows the analytical discipline emphasized by Kass et al., 2016.

5.2 Attack Mechanisms and Documented Data Exfiltration:

Table 5 presents the conditional proportions addressing the three research questions. Differences compare the exfiltration proportion among records coded one for a mechanism with the proportion among records coded zero.

Table 5: Documented exfiltration by attack mechanism

Mechanism	Incidents With Mechanism Coded	Exfiltration Records Within This Group	Exfiltration Percentage	Incidents Without Mechanism Coded	Exfiltration Records Within This Group	Exfiltration Percentage	Difference in Percentage Points
Account access	34	22	64.71	141	116	82.27	17.56 lower
Self signed or unsigned software	85	64	75.29	90	74	82.22	6.93 lower
Stolen or purchased certificates	25	20	80.00	150	118	78.67	1.33 higher

Source: Calculations from 175 incident records. Mechanism categories can overlap. Differences are unadjusted descriptive comparisons.

For RQ1, account access records exhibit a lower proportion of documented exfiltration. The outcome appears in 22 of 34 incidents with coded account access, compared with 116 of 141 incidents without that coding. The difference is 17.56 percentage points, the largest absolute contrast among the three mechanisms. This result separates access to software modification permissions from the eventual recorded impact. It does not indicate that account compromise protects information. Rather, account access identifies an entry mechanism whose observed outcomes differ across incidents. Rose et al., 2020, distinguish authentication from authorization and reject implicit trust based on account ownership. The present comparison adds evidence about recorded outcomes without measuring the effectiveness of those security principles.

For RQ2, documented exfiltration appears in 64 of 85 incidents involving self-signed or unsigned software, compared with 74 of 90 incidents without that coding. The corresponding difference is 6.93 percentage points lower. Exfiltration remains common in both groups, so the contrast reflects a difference in proportions rather than a separation between exfiltration and other attack objectives.

For RQ3, certificate acquisition records show similar exfiltration proportions in the full sample. Twenty of 25 incidents involving stolen or purchased certificates contain documented exfiltration, compared with 118 of 150 other incidents. The difference is 1.33 percentage points higher. The certificate comparison concerns misuse of signing authority. It does not evaluate whether legitimate code signing prevents information theft. Cooper et al., 2018, explain that signatures establish integrity and signer authentication. The observed exfiltration records demonstrate why those functions must remain conceptually distinct from the behavior of the signed software.

5.3 Sensitivity to incomplete impact information:

Nine incidents carry an unknown or other impact flag. Four also contain documented exfiltration, while five do not. Excluding all nine therefore changes both the sample composition and the distribution of the outcome. The resulting sample contains 166 incidents, including 134 exfiltration records, giving an overall proportion of 80.7 percent.

Table 6: Mechanism comparisons after excluding unknown or other impact records

Mechanism	Exfiltration Records Among Coded Incidents	Exfiltration Percentage	Exfiltration Records Among Other Incidents	Exfiltration Percentage	Difference in Percentage Points
Account access	21 of 31	67.74	113 of 135	83.70	15.96 lower
Self signed or unsigned software	64 of 82	78.05	70 of 84	83.33	5.28 lower
Stolen or purchased certificates	16 of 18	88.89	118 of 148	79.73	9.16 higher

The account access contrast remains negative, changing from 17.56 to 15.96 percentage points lower. The self signed or unsigned contrast also remains negative, changing from 6.93 to 5.28 percentage points lower. Their directions persist under the stricter reporting criterion, although the magnitudes decrease. The certificate contrast is more sensitive. Its difference increases from 1.33 to 9.16 percentage points higher. Seven of the nine excluded incidents involve stolen or purchased certificates, reducing that group from 25 to 18 records. Four of these seven excluded certificate incidents lack coded exfiltration. Their removal explains the increase in the conditional exfiltration proportion. This sensitivity identifies a reporting composition issue rather than confirming a stronger certificate association. Exclusion based on impact completeness can select incidents with different outcome profiles. Greenland et al., 2016, emphasize that statistical interpretation depends on analytical assumptions and selection decisions. Here, the material change in magnitude requires retaining both sample definitions in the interpretation.

5.4 Integrated Answer to the Research Questions:

The descriptive evidence answers the research questions with three distinct patterns. Account access has the largest negative contrast in documented exfiltration, self-signed or unsigned software has a smaller negative contrast, and certificate acquisition has a small positive contrast that increases after excluding incomplete impact records. The findings show that attack mechanisms do not map uniformly onto documented exfiltration. Their relevance lies in distinguishing the route used to compromise a software supply chain from the impact recorded within that incident. Hammi et al., 2023, describe security threats across interconnected supply chain technologies. The present analysis complements that system perspective by showing variation in a specific recorded outcome across three mechanisms. These conclusions apply to the selected incident collection. Overlapping mechanisms, retrospective reporting and the meaning of uncoded outcomes constrain interpretation. The account access and self-signed or unsigned comparisons retain their direction under the completeness check, whereas the certificate comparison depends more strongly on which impact records are retained.

6. Discussion:

An attack's entry mechanism does not determine its documented impact. Table 5 shows that account access has the largest negative contrast in recorded exfiltration, self-signed or unsigned software has a smaller negative contrast, and certificate acquisition has little separation in the full sample. These differences challenge the use of attack categories as interchangeable indicators of information theft. Andreoli et al., 2023, investigate characteristic malicious behaviors within software supply chain attacks, emphasizing what compromised software does. This analysis adds a complementary distinction between the mechanism recorded for an incident and its documented exfiltration outcome. Its contribution is an outcome specific comparison across three mechanisms, rather than a general ranking of attack severity. The negative contrasts describe differences among reported attacks and do not imply that account compromise or unsigned software reduces exposure. The findings refine the trust boundary explanation by separating opportunities for compromise from evidence of downstream information loss. Account access concerns modification permissions, while self-signed or unsigned software concerns the assurance attached to an artifact. Neither category directly measures execution privileges, access to sensitive information or the behavior of a payload. The coexistence of exfiltration and other outcomes within these categories therefore matters theoretically: crossing a trust boundary is an enabling condition whose consequences remain context dependent. Ponta et al., 2020, demonstrate the value of examining whether vulnerable code is reachable within a particular application rather than relying exclusively on dependency metadata. Table 5 extends this distinction to incident classification by showing that an entry category alone provides an incomplete account of documented exfiltration. The framework consequently requires separate treatment of compromise mechanisms and observed consequences.

Mechanism frequency also differs from the magnitude of its outcome contrast. Table 4 identifies self signed or unsigned software as the most frequent selected mechanism, yet Table 5 shows a larger absolute exfiltration contrast for account access. Certificate acquisition is less frequent and produces the smallest full sample contrast. These patterns prevent interpreting prevalence as pathway dominance. Moreover, overlapping categories mean that their counts cannot be decomposed into independent shares of exfiltration. Zerouali et al., 2022, distinguish vulnerable packages from the direct and indirect dependents exposed through dependency networks. That distinction concerns exposure rather than realized information theft, but it reinforces the need to specify which stage of risk an indicator represents. This analysis contributes a corresponding separation between mechanism prevalence and conditional outcome proportions, preventing frequent attack features from being mistaken for dominant explanations of exfiltration. Reporting completeness changes the interpretation of certificate related incidents more than the other comparisons. Table 6 shows that the certificate contrast increases from approximately one to nine percentage points after excluding records with unknown or other impact information. Seven of the nine excluded incidents involve certificates, making the restriction uneven across mechanisms. The resulting change reveals how evidence availability can alter an apparent relationship without any change in the underlying attack process. Chinthanet et al., 2021, identify distinct stages in the release, adoption and propagation of vulnerability fixes, showing why software security processes require stage specific measurement. Here, the relevant distinction is between an attack characteristic and the completeness of its documented impact. The contribution

is to show that reporting composition belongs within the interpretation of mechanism comparisons, particularly when a small subgroup loses a substantial share of its records.

The international relevance lies in shared software relationships rather than a demonstrated difference between national economies. Decan et al., 2019, identify both similarities and differences across seven packaging ecosystems, including concentrated dependencies and growing network fragility. The present findings complement that structural perspective by distinguishing the possibility of transmitting compromise from the outcome documented within an incident. Tables 5 and 6 suggest that security teams should retain separate fields for attack mechanisms, observed impacts and impact completeness rather than use a single category to rank exfiltration exposure. In particular, the lower account access proportion should not justify reducing account protection, while the unstable certificate contrast warrants closer incident review before prioritization. The theoretical refinement is therefore a separation of transmission opportunity, documented consequence and evidence completeness, allowing global software security comparisons to remain sensitive to what their underlying records actually measure.

7. Conclusion and Implications:

Global software security depends on distinguishing how compromise occurs from what an incident demonstrably reveals. This analysis shows that different routes across software trust boundaries exhibit distinct patterns of documented information loss, while incomplete impact reporting can materially alter those comparisons. Its contribution is an empirical distinction among compromise opportunities, recorded consequences and evidence completeness. This distinction refines the trust boundary perspective by showing why an entry mechanism cannot substitute for an outcome assessment. The findings support treating upstream authority and downstream harm as separate analytical stages, without attributing causal effects to the observed contrasts. For managers, this means assessing incidents through documented consequences rather than ranking exposure solely by attack labels or their frequency. Security teams should maintain separate records of compromise routes, observed impacts and uncertainty, then review sensitive classifications before allocating investigative resources. For policymakers and industry bodies, the evidence supports clearer incident reporting standards that distinguish confirmed observations from uncoded information and preserve source traceability. Operationally, these practices would make comparisons across organizations and software ecosystems more interpretable. Their broader value lies in supporting accountable security decisions where software relationships cross institutional and geographical boundaries. Better evidence classification can help organizations communicate exposure accurately without converting uncertain observations into confident assurances.

The incident collection offers opportunities to strengthen outcome measurement and generalizability through independently adjudicated records. Future research should distinguish confirmed absence from incomplete reporting and link related incidents through campaign and infrastructure identifiers. Longitudinal evidence could establish the sequence connecting initial compromise, execution and information transfer. Comparisons across software ecosystems could assess whether the descriptive patterns persist under consistent reporting rules. Linking incident records to deployment exposure and security practices would support stronger tests of competing explanations. These extensions would clarify when the observed contrasts reflect attacker objectives, technical permissions or documentation processes, and identify which insights transfer across settings.

References:

1. Andreoli, A., Lounis, A., Debbabi, M., & Hanna, A. 2023. On the prevalence of software supply chain attacks: Empirical study and investigative framework. Forensic Science International: Digital Investigation, volume 44, supplement, Article 301508. https://doi.org/10.1016/j.fsidi.2023.301508
2. Atlantic Council. 2020. Breaking trust: Shades of crisis across an insecure software supply chain: Database codebook. https://www.atlanticcouncil.org/wp-content/uploads/2020/07/Breaking-Trust-Database-Codebook.pdf
3. Boyens, J., Smith, A., Bartol, N., Winkler, K., Holbrook, A., & Fallon, M. 2022. Cyber security supply chain risk management practices for systems and organizations. NIST Special Publication 800-161, Revision 1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-161r1
4. Chinthanet, B., Kula, R. G., McIntosh, S., Ishio, T., Ihara, A., & Matsumoto, K. 2021. Lags in the release, adoption, and propagation of npm vulnerability fixes. Empirical Software Engineering, volume 26, issue 3, Article 47. https://doi.org/10.1007/s10664-021-09951-x
5. Cooper, D., Regenscheid, A., Souppaya, M., Bean, C., Boyle, M., Cooley, D., & Jenkins, M. 2018. Security considerations for code signing. NIST Cyber security White Paper 5. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.5
6. Decan, A., Mens, T., & Grosjean, P. 2019. An empirical comparison of dependency network evolution in seven software packaging ecosystems. Empirical Software Engineering, volume 24, issue 1, pages 381-416. https://doi.org/10.1007/s10664-017-9589-y
7. Goodman, A., Pepe, A., Blocker, A. W., Borgman, C. L., Cranmer, K., Crosas, M., Di Stefano, R., Gil, Y., Groth, P., Hedstrom, M., Hogg, D. W., Kashyap, V., Mahabal, A., Siemiginowska, A., &Slavkovic, A. 2014. Ten simple rules for the care and feeding of scientific data. PLOS Computational Biology, volume 10, issue 4, Article e1003542. https://doi.org/10.1371/journal.pcbi.1003542
8. Greenland, S., Senn, S. J., Rothman, K. J., Carlin, J. B., Poole, C., Goodman, S. N., & Altman, D. G. 2016. Statistical tests, P values, confidence intervals, and power: A guide to misinterpretations. European Journal of Epidemiology, volume 31, issue 4, pages 337-350. https://doi.org/10.1007/s10654-016-0149-3
9. Hammi, B., Zeadally, S., & Nebhen, J. 2023. Security threats, countermeasures, and challenges of digital supply chains. ACM Computing Surveys, volume 55, issue 14s, Article 316, pages 1-40. https://doi.org/10.1145/3588999
10. Kass, R. E., Caffo, B. S., Davidian, M., Meng, X. L., Yu, B., & Reid, N. 2016. Ten simple rules for effective statistical practice. PLOS Computational Biology, volume 12, issue 6, Article e1004961. https://doi.org/10.1371/journal.pcbi.1004961

11. Loomis, W., Scott, S., Herr, T., Brackett, S. A., Messieh, N., & Lee, J. 2023. Software supply chain security: The dataset. Atlantic Council. https://www.atlanticcouncil.org/content-series/Cyber security-policy-and-strategy/software-supply-chain-security-the-dataset/
12. Ponta, S. E., Plate, H., & Sabetta, A. 2020. Detection, assessment and mitigation of vulnerabilities in open source dependencies. Empirical Software Engineering, volume 25, issue 5, pages 3175-3215. https://doi.org/10.1007/s10664-020-09830-x
13. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. 2020. Zero trust architecture. NIST Special Publication 800-207. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207
14. Souppaya, M., Scarfone, K., & Dodson, D. 2022. Secure software development framework version 1.1: Recommendations for mitigating the risk of software vulnerabilities. NIST Special Publication 800-218. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-218
15. Zerouali, A., Mens, T., Decan, A., & De Roover, C. 2022. On the impact of security vulnerabilities in the npm and RubyGems dependency networks. Empirical Software Engineering, volume 27, issue 5, Article 107. https://doi.org/10.1007/s10664-022-10154-1