International Journal of Advanced Trends in Engineering and Technology (IJATET)
Impact Factor: 5.965, ISSN (Online): 2456 - 4664
(www.dvpublication.com) Volume 7, Issue 2, 2022

VULNERABILITY TYPES AND SEVERITY IN FRAPPE FRAMEWORK AND ERPNEXT SOFTWARE

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, "Vulnerability Types and Severity in Frappe Framework and ERPNext Software", International Journal of Advanced Trends in Engineering and Technology, Volume 7, Issue 2, Page Number 105-110, 2022.

Abstract:

This analysis examines how the assessed severity of software vulnerabilities varies across weakness types in open-source products used internationally. It uses six selected vulnerability records for Frappe Framework and ERPNext, published in 2019 and 2020, and compares their recorded severity scores through descriptive analysis. The two SQL injection records each received a score of 8.8, while the two cross site scripting records each received 6.1. A server-side template injection record received the highest individual score, at 9.8. The observed differences describe the selected records and do not establish a general ranking of weakness types or a causal relationship. The comparison highlights the need to interpret a severity score alongside the technical conditions of the individual vulnerability rather than infer severity from its category alone. For software maintainers and security teams, the findings support examining advisory details when prioritizing review. Broader sampling is needed to determine whether the observed pattern persists across products and contexts.

Key Words: Descriptive Analysis; Open-Source Software; Security Assessment; Vulnerability Severity; Vulnerability Types

1. Introduction:

Software vulnerabilities can expose applications and their users to unauthorized actions, but the nature of a weakness does not by itself establish the severity of every instance. This distinction matters when maintainers must interpret disclosed vulnerabilities and decide which issues require technical attention. Vulnerability records provide descriptions and severity assessments that support such decisions, although a severity score alone does not establish whether a vulnerability will be exploited or what losses will occur (Allodi et al., 2020; Jacobs et al., 2020).

The relationship between weakness type and severity depends on the conditions under which the weakness can affect an application. Injection weaknesses, for example, can allow user-controlled data to reach operations that affect a database or content displayed to users. These operations have different technical consequences, but the impact of an individual vulnerability depends on more than its category (Spoto et al., 2019). Vulnerability descriptions can help estimate the characteristics used in severity assessment, as demonstrated by Spanos and Angelis (2018). Their work supports examining the information contained in descriptions without assuming that every record in a category has the same score.

Severity measurement also requires careful interpretation. The Common Vulnerability Scoring System provides a standardized base score for comparing assessed vulnerability characteristics. Expert judgments can nevertheless differ from assigned scores, and assessment accuracy can depend on the information and expertise available to evaluators (Holm & Afridi, 2015; Allodi et al., 2020). A comparison of scores therefore concerns assessed technical severity. It cannot, without additional evidence, establish exploitation, organizational loss, or the effectiveness of remediation.

Existing research has linked vulnerability records to source code and fixes across open-source projects (Bhandari et al., 2021). Other work has examined vulnerability characteristics and severity assessment across larger collections of records (Spanos & Angelis, 2018). These studies provide methods and context for examining disclosed weaknesses, but their findings do not establish how severity scores vary across the particular records selected for Frappe Framework and ERPNext. A focused comparison can document that variation while recognizing that a small selected sample cannot determine whether its pattern extends to all vulnerabilities in either product.

This study aims to examine how recorded vulnerability severity varies across the vulnerability types identified in available records for Frappe Framework and ERPNext. The research question is: How do recorded vulnerability severity scores vary across the vulnerability types identified in the available Frappe Framework and ERPNext records? The analysis uses six vulnerability records published in 2019 and 2020, with each record serving as one observation. It compares reported weakness descriptions with their recorded base severity scores using descriptive statistics.

The setting provides a comparison of vulnerabilities associated with two related open-source products maintained by an Indian software organization. The findings provide a basis for examining whether scores differ among the selected weakness types; they do not establish a causal relationship or a general ranking of

vulnerability categories. For maintainers and researchers working in other settings, the comparison illustrates the value of examining individual severity assessments alongside weakness descriptions. Its transferability remains limited by the small sample, the related products, and the absence of a documented complete sampling frame.

2. Theory, Conceptual Framework and Research Question:

Software vulnerability analysis considers how weaknesses in an application may expose its functions and data to unauthorized actions. In an open-source system, maintainers, security researchers, and users depend on information about disclosed weaknesses to assess their potential consequences. Bhandari et al., 2021, demonstrate how vulnerability records and source code information can be linked to support analysis of disclosed software weaknesses.

This analysis examines vulnerability type and vulnerability severity in the available records for Frappe Framework and ERPNext. Vulnerability type identifies the weakness described in each record. Vulnerability severity is represented by the recorded Common Vulnerability Scoring System base score. These are observed characteristics of disclosed vulnerabilities, rather than latent constructs. The framework considers whether the recorded severity scores vary across vulnerability types.

The mechanism depends on how a weakness permits an attacker to interact with an application. Spoto et al., 2019, explain that injection weaknesses can arise when user-controlled data reach application operations without sufficient restriction. SQL injection can affect database operations, while cross site scripting can affect content presented to users. These mechanisms provide a basis for expecting differences in potential impact, but a weakness category alone does not establish the severity of every vulnerability within it.

Severity scores reflect an assessment of vulnerability characteristics. Spanos and Angelis, 2018, show that vulnerability descriptions can provide information for estimating those characteristics and associated scores. However, Holm and Afridi, 2015, find that expert judgments can differ from standardized base scores. The score therefore represents assessed technical severity; it does not establish that vulnerability was exploited or measure the harm experienced by a particular user.

The framework treats the relationship between vulnerability type and severity as an association. It does not assume that the weakness label alone causes a particular score. Gueye and Mell, 2021, show that vulnerability characteristics and scores can be examined as distributions, supporting a descriptive comparison of the available records.

The six available records permit an exploratory comparison, but their small number and limited coverage do not support a reliable directional hypothesis test. The analysis is therefore guided by the following research question: How do recorded vulnerability severity scores vary across the vulnerability types identified in the available Frappe Framework and ERPNext records?

3. Data:

This analysis uses vulnerability records to examine vulnerability type and assessed severity in Frappe Framework and ERPNext. The section identifies the data sources, sample, observed variables, and limits of the descriptive comparison.

3.1 Data Sources, Population, and Sample Construction:

The analysis uses six records from the National Vulnerability Database, maintained by the National Institute of Standards and Technology. Each record describes a disclosed vulnerability and reports an assessed severity score. The workbook identifies each record through a unique vulnerability identifier and provides a source link. The source records include vulnerabilities published in 2019 and 2020 (National Institute of Standards and Technology, 2019, 2020).

The empirical setting comprises Frappe Framework and ERPNext, two related open source software products associated with Frappe Technologies in India. The unit of analysis is an individual vulnerability record. The available records form a collection of vulnerability observations rather than a panel of products observed at regular intervals.

The target population comprises disclosed vulnerabilities affecting the two products during 2019 and 2020. The accessible sample consists of the six records in the supplied workbook. The workbook does not establish the total number of eligible vulnerabilities, the completeness of the search, or the rules used to exclude other records. The six observations are therefore treated as a selected sample.

Table 1: Sample construction

Stage	Software Products	Vulnerability Records	Basis
Target population	2	Unknown	Disclosed vulnerabilities affecting the two products during 2019 and 2020
Accessible workbook sample	2	6	Records included in the supplied workbook
Descriptive comparison	2	6	Records contain a weakness description and a severity score

The workbook records current severity assessments for historical disclosures. The publication year identifies when a vulnerability was disclosed. It does not establish that the recorded score was available in the same form at that time. The workbook does not identify a database release or licence.

3.2 Data Integration, Cleaning, and Quality Assurance:

The workbook presents the records in a single table. The vulnerability identifier distinguishes each observation and links it to its source record. The six identifiers are distinct. The documentation does not establish whether records were merged from multiple files, whether candidate records were excluded, or whether source values conflicted during collection.

All six records contain a weakness description and a severity score. Two records have no verified common weakness identifier, while one carries a source label indicating insufficient weakness information. These gaps do not prevent the descriptive comparison because it uses the recorded weakness descriptions. They limit analysis based exclusively on common weakness identifiers.

The workbook does not document imputation, interpolation, outlier removal, or changes to severity scores. The records contain scores from versions 3.0 and 3.1 of the Common Vulnerability Scoring System. Assessment differences can affect how severity is represented. Holm and Afridi (2015) found that expert judgments can differ from standardized base scores.

3.3 Variable Measurement, Construction, and Validation:

The analysis uses two observed variables. Vulnerability type is a categorical description of the weakness. Vulnerability severity is the numerical base score recorded in the workbook. The comparison uses the reported scores without normalization, weighting, or construction of a composite index.

Table 2: Variable dictionary

Variable	Analytical Role	Definition and Source	Unit and Coverage	Missing Value Rule and Limitation
Vulnerability type	Descriptive grouping variable	Weakness description recorded for each vulnerability in the National Vulnerability Database	Category for each vulnerability; six records	All six records have descriptions. Some common weakness identifiers are unavailable or insufficiently specified
Vulnerability severity	Outcome	Recorded Common Vulnerability Scoring System base score from the National Vulnerability Database	Score from zero to ten; six records	No scores are missing. The sample includes scores from versions 3.0 and 3.1

A higher score represents higher assessed base severity. It does not directly measure actual exploitation or losses experienced by users. Spanos and Angelis (2018) explain that vulnerability characteristics inform severity assessment. Spoto et al. (2019) identify mechanisms through which different injection weaknesses can affect software operations.

Across the six records, the mean severity score is 7.85, the sample standard deviation is 1.54, and the median is 8.15. Scores range from 6.1 to 9.8. These statistics describe the selected records only. The workbook does not document an independent validation exercise or establish that the sample represents all vulnerabilities affecting either product.

3.4 Analytical Framework, Identification, and Reproducibility:

The analysis is descriptive and exploratory. It compares recorded severity scores across the vulnerability types described in the six records. The comparison can identify the scores observed within each type, but it cannot establish that vulnerability type causes severity or that the observed pattern applies to other software.

The sample is too small to support a reliable multivariable model or a generalizable statistical test. Some vulnerability types appear only once, which prevents assessment of variation within those types. The analysis therefore uses no regression, controls, interactions, or causal identification strategy.

The workbook supplies record identifiers, source links, descriptions, and scores that allow the descriptive comparison to be traced to its documented inputs. It does not provide a complete search log, a historical database snapshot, or construction code. These gaps limit assessment of selection bias and independent reproduction of the collection process.

4. Methods:

4.1 Research Design and Analytical Objective:

The analysis uses a descriptive design to examine how recorded severity scores vary across vulnerability types in Frappe Framework and ERPNext. The unit of analysis is an individual vulnerability record. The analytical objective is to describe the scores observed within each recorded weakness category and compare the categories represented in the sample.

The design examines differences among selected records. It does not identify the causal effect of vulnerability type on severity. Differences in individual vulnerability characteristics and assessment judgments may also contribute to the recorded scores (Allodi et al., 2020).

4.2 Sample and Measurement Alignment:

The analytical sample contains six vulnerability records published in 2019 and 2020. Each record has a weakness description and a severity score. The sample construction table and variable dictionary identify the two software products, the recorded measures, and their coverage.

Vulnerability type is treated as a categorical grouping variable based on the description in the workbook. Severity is measured using the recorded Common Vulnerability Scoring System base score. The analysis retains the reported scores and identifies their scoring versions. The score describes assessed technical severity rather than observed exploitation or loss (Holm & Afridi, 2015).

The records form a selected sample. The workbook does not establish the total number of eligible disclosures or the completeness of the search. The comparison is therefore limited to the records available.

4.3 Data Processing and Descriptive Analysis:

The six records are grouped according to their recorded weakness descriptions. The analysis reports the number of records and the individual severity scores within each group. It calculates the arithmetic mean score for each group and summarizes the full sample using the mean, median, sample standard deviation, minimum, and maximum.

All six records contain the two measures used in the comparison. Missing common weakness identifiers are not replaced with zero or imputed. The workbook does not document outlier removal or changes to the recorded scores. No regression model or interaction term is used.

4.4 Interpretation and Reproducibility:

The comparison is exploratory. Differences between group summaries describe the selected records and are not interpreted as causal effects or population estimates. Two weakness categories contain one record each, so their within category variation cannot be assessed. No inferential test is used to generalize the observed differences.

The workbook provides vulnerability identifiers and source links that permit readers to trace the reported descriptions and scores. It does not provide a complete search log, a historical database snapshot, or an analysis script. These gaps limit independent reproduction of the original record selection. The descriptive calculations can be reproduced from the six recorded scores and their weakness descriptions.

5. Findings:

This analysis examines how recorded vulnerability severity varies across weakness types in Frappe Framework and ERPNext. The findings address the research question through a descriptive comparison of six vulnerability records. The evidence establishes patterns within this selected sample and does not support causal or generalizable conclusions.

The unit of analysis is an individual vulnerability record. Four records were published in 2019 and two in 2020. Five concern Frappe Framework and one concerns ERPNext. All six contain a weakness description and a severity score. The sample does not establish the total number of eligible vulnerabilities or the completeness of the original search.

Table 1: Recorded vulnerability severity by type

Vulnerability Type	Records	Severity Scores	Mean Score
SQL injection	2	8.8, 8.8	8.8
Cross site scripting	2	6.1, 6.1	6.1
Server side template injection	1	9.8	9.8
Two factor authentication secret exposure	1	7.5	7.5
All recorded vulnerabilities	6	6.1 to 9.8	7.85

Note. Scores are Common Vulnerability Scoring System base scores recorded in the supplied workbook. The table includes scores from versions 3.0 and 3.1.

The two SQL injection records each have a severity score of 8.8. The two cross site scripting records each have a score of 6.1. The observed difference between these group means is 2.7 points. This comparison is consistent with the distinction between weaknesses affecting database operations and weaknesses affecting content presented to users, although the recorded type alone does not determine the score of every vulnerability within a category (Spoto et al., 2019).

The server-side template injection record has the highest score, at 9.8. The authentication secret exposure record has a score of 7.5. Each type appears once, so the sample provides no evidence about variation within either category. The scores represent assessed base severity rather than observed exploitation or realized losses. Expert assessments can also differ from standardized scores, which limits interpretation of the scores as complete measures of operational risk (Holm & Afridi, 2015).

Across all six records, the mean score is 7.85, the median is 8.15, and the sample standard deviation is 1.54. The scores range from 6.1 to 9.8. The six weakness descriptions and six scores are available for the descriptive comparison. Two records lack a verified common weakness identifier, and one has a label indicating insufficient weakness information.

The findings answer the research question for the available records: severity scores vary across the recorded vulnerability types. The SQL injection records have higher scores than the cross-site scripting records, while the highest individual score belongs to a server-side template injection record. The small selected sample and single observations in two categories prevent a reliable test of whether these differences extend beyond the records examined.

6. Discussion:

A vulnerability category does not carry a single severity value across all software contexts. In Table 1, the two SQL injection records have higher scores than the two cross site scripting records. This difference describes the selected vulnerabilities, but it does not establish a general ranking of the two weakness types. The contribution of the analysis is a documented comparison within Frappe Framework and ERPNext; the available evidence does not establish that this comparison is unprecedented.

The contrast is compatible with the view that severity depends on the characteristics of an individual vulnerability as well as its weakness type. Allodi et al. (2020) found that vulnerability assessment requires judgments about technical conditions and that assessment accuracy varies across evaluators. The records examined here provide severity scores and weakness descriptions, but they do not measure the intermediate processes needed to establish why the scores differ.

The highest individual score belongs to the server-side template injection record. Its presence shows that the observed range is not explained solely by the comparison between SQL injection and cross site scripting. However, each of the other two weakness types appears only once. The evidence cannot establish which type generally dominates or whether the observed ordering would persist in a larger sample.

The distinction between assessed severity and operational risk also matters for interpretation. Jacobs et al. (2020) examine exploitation in real environments and show the value of additional information for prioritizing remediation. The present records contain no exploitation or loss measure. A higher base score can inform technical review, but it cannot demonstrate that the corresponding vulnerability caused greater harm in an operational setting.

The comparison offers a focused view of disclosed vulnerabilities associated with an Indian software organization, while its implications remain relevant to users and maintainers of open source systems elsewhere. The practical implication is to examine the conditions and affected components behind each score rather than relying on weakness labels alone. Further research can assess whether the observed pattern persists by collecting a documented, broader set of vulnerability records and comparing severity within types across products and assessment conditions.

7. Conclusion and Implications:

Open-source software security depends on understanding the conditions behind a vulnerability, not only its category. This analysis found differences in assessed severity among the selected records for Frappe Framework and ERPNext. Its contribution is a focused comparison that connects disclosed weakness descriptions with their recorded severity assessments. The findings support an interpretation of severity as a property assessed for an individual vulnerability rather than a fixed attribute of a weakness category. They do not establish why the scores differ or whether the observed pattern applies across other products. For maintainers and security teams, the practical implication is to review the affected component and the conditions described in each advisory when assessing a vulnerability. Severity scores can inform this review, but decisions about remediation also require information about exposure and the operating environment. The analysis offers a relevant question for open-source communities in different settings: whether descriptive patterns observed in a small set of disclosures persist when examined across a broader range of software.

The six records provide an opportunity for a more comprehensive investigation. Future research can construct a documented sample of disclosures across additional projects and compare cases within the same weakness categories. Consistent scoring versions and historical snapshots would improve assessment of changes over time. Linking vulnerability records to evidence on affected deployments, fixes, and exploitation would also allow researchers to examine practical consequences. Such extensions could assess whether the observed differences recur across products and contexts without treating the present comparison as evidence of causality.

References:

1. Allodi, L., Cremonini, M., Massacci, F., & Shim, W. (2020). Measuring the accuracy of software vulnerability assessments: Experiments with students and professionals. Empirical Software Engineering, 25(2), 1063-1094. https://doi.org/10.1007/s10664-019-09797-4
2. Bhandari, G. P., Naseer, A., & Moonen, L. (2021). CVEfixes: Automated collection of vulnerabilities and their fixes from open-source software. In Proceedings of the 17th International Conference on

Predictive Models and Data Analytics in Software Engineering (pp. 30-39). Association for Computing Machinery. https://doi.org/10.1145/3475960.3475985

3. Gueye, A., & Mell, P. (2021). A historical and statistical study of the software vulnerability landscape. In The Seventh International Conference on Advances and Trends in Software Engineering (pp. 1-7). https://www.nist.gov/publications/historical-and-statistical-study-software-vulnerability-landscape
4. Holm, H., & Afridi, K. K. (2015). An expert-based investigation of the Common Vulnerability Scoring System. Computers & Security, 53, 18-30. https://doi.org/10.1016/j.cose.2015.04.012
5. Jacobs, J., Romanosky, S., Adjerid, I., & Baker, W. (2020). Improving vulnerability remediation through better exploit prediction. Journal of Cybersecurity, 6(1), Article tyaa015. https://doi.org/10.1093/cybsec/tyaa015
6. National Institute of Standards and Technology. (2019). CVE-2019-14967. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2019-14967
7. National Institute of Standards and Technology. (2020). CVE-2020-6145. National Vulnerability Database. https://nvd.nist.gov/vuln/detail/CVE-2020-6145
8. Spanos, G., & Angelis, L. (2018). A multi-target approach to estimate software vulnerability characteristics and severity scores. Journal of Systems and Software, 146, 152-166. https://doi.org/10.1016/j.jss.2018.09.039
9. Spoto, F., Burato, E., Ernst, M. D., Ferrara, P., Lovato, A., Macedonio, D., & Spiridon, C. (2019). Static identification of injection attacks in Java. ACM Transactions on Programming Languages and Systems, 41(3), Article 18, 1-58. https://doi.org/10.1145/3332371