Analysis and Implementation of SPARTA-to-CWE Mapping with Weighted Set Cover Optimisation

Reantric

September 8, 2024

Abstract

This report provides an in-depth analysis of the SPARTA framework and its mapping to the Common Weakness Enumeration (CWE) vulnerabilities, with an emphasis on applying a Weighted Set Cover Optimisation approach. The report discusses data extraction, the creation of mappings between SbD, SPARTA, and CWE entities, and the design of a visualisation tool to display this relationship dynamically. The report concludes with observations about the efficiency and scalability of the solution in visualising large datasets.

Contents

1 Introduction

The objective of this project is to analyse the relationship between SPARTA Identifiers and Common Weakness Enumerations (CWEs) while proposing a strategy to mitigate these vulnerabilities through the Security by Design (SbD) framework. This report explores how we leveraged weighted set cover algorithms and visualisation tools to represent and optimise these mappings.

1.1 SPARTA and CWEs

SPARTA is a cybersecurity taxonomy that categorises a variety of threats, vulnerabilities, and security strategies. CWEs, on the other hand, are standardised types of weaknesses commonly found in software. Each SPARTA ID is associated with a set of CWEs that represent specific security concerns.

The task involved identifying CWEs linked to each SPARTA ID and finding effective SbD strategies to mitigate them. These relationships were extracted from a dataset and optimised using weighted set cover algorithms.

2 Data Extraction and Transformation

2.1 SPARTA-to-CWE Mapping

The initial step was to extract data from the SPARTA dataset and map the SPARTA IDs to their associated CWEs. The SPARTA sheet contained information about which CWEs were related to a particular SPARTA identifier, and the corresponding weaknesses had to be carefully organised for further analysis.

- **SPARTA Sheet:** Contains SPARTA IDs and their corresponding CWEs.
- **CWE Sheet:** Lists the vulnerabilities and their details.
- **SbD Mapping:** Maps SbD strategies to relevant CWEs.

The dataset was processed using Python for data extraction and transformed into usable mappings for the Set Cover problem.

2.2 Generating SbD Mappings

Each SbD Identifier was mapped to one or more CWEs, reflecting the vulnerabilities it could mitigate. This mapping was key to constructing the sets that would be used in the Set Cover Optimisation.

The SbD-to-CWE map was created using the following logic:

- Parse the SbD Sheet to create an initial set of SbD identifiers.
- For each CWE in the dataset, associate it with relevant SbDs based on their relationship.

3 Weighted Set Cover Optimisation

3.1 Problem Definition

The Weighted Set Cover problem is defined as selecting the smallest subset of SbD strategies that collectively mitigate all identified CWEs, while also minimising the cost associated with these strategies. In this context, each SbD strategy has a weight (or cost) that indicates its importance or resource demand in the overall system.

The optimisation was modelled as follows:

- Each SbD strategy is treated as a "set" containing the CWEs it covers.
- Each SbD strategy has a weight that influences the selection process.
- The objective is to minimise the total weight of the SbD strategies selected while ensuring all CWEs are covered (if possible).

Mathematically, this can be formulated as:

$$\text{minimise} \sum_{i \in \text{SbDs}} w_i x_i \quad \text{subject to} \quad \sum_{i \in \text{SbDs}} a_{ij} x_i \geqslant 1, \quad \forall j \in \text{CWEs}$$

where x_i is a binary variable indicating whether SbD i is selected, w_i is the weight of SbD i, and a_{ij} indicates whether SbD i covers CWE j.

3.2 Algorithm Implementation

The Weighted Set Cover algorithm relies on two approaches: a dynamic programming (DP) solution for smaller cases and a greedy heuristic for larger cases. The choice between these approaches is determined by whether the size of the search space $|\text{SbDs}| \cdot 2^{|\text{CWEs}|}$ exceeds a certain threshold.

If the threshold of 10^7 is exceeded, the greedy algorithm is used to ensure efficiency in processing larger datasets. Otherwise, the more precise dynamic programming approach is employed. This combination ensures a balance between computational efficiency and solution quality.

3.2.1 Dynamic Programming Approach

For smaller datasets, the dynamic programming solution guarantees an exact optimal solution for the weighted set cover problem. The steps are as follows:

1. A state is maintained for each subset of CWEs, representing the minimum cost to cover that subset.
2. For each SbD strategy, the algorithm checks which CWEs can be covered and updates the state.
3. This is repeated for all possible subsets of CWEs, ensuring that the minimum cost set cover is found.

This approach ensures that the smallest possible subset of SbD strategies is selected, but it has a time complexity of $O(m \cdot 2^n)$, where m is the number of SbDs and n is the number of CWEs. Therefore, it is only feasible for smaller datasets.

3.2.2 Greedy Heuristic Approach

For larger datasets, where the dynamic programming approach is computationally prohibitive, a greedy heuristic is used. The steps are as follows:

1. For each SbD strategy, determine the set of CWEs it can mitigate and calculate a cost-efficiency ratio ($\frac{\text{coverage}}{\text{weight}}$).
2. At each step, select the SbD strategy with the highest cost-efficiency ratio.
3. Continue until all CWEs have been mitigated.

This greedy approach guarantees a near-optimal solution[1] for the weighted set cover problem. It ensures scalability for large datasets while providing solutions that are close to optimal.

4 Visualisation Tool Design

4.1 Graph Representation

The graph-based visualisation tool was designed using p5.js. It represents:

- **Nodes:** Representing CWEs and SbDs.
- **Edges:** Connecting SbDs to CWEs they mitigate.

The tool also supports context menus, dynamic zooming, customisable search, and camera panning to efficiently navigate large datasets.

[1] More specifically, it guarantees a $\Theta(\log n)$ approximation ratio solution, which is believed to be the best possible unless $P = NP$.

4.2 Zoom and Camera Adjustments

Due to the large number of CWEs and SbDs, the visualisation tool incorporated zooming and panning functionalities to allow users to explore the relationships between SbDs, SPARTA IDs, and CWEs effectively.

5 Challenges and Performance Optimisations

During the course of this project, over 9,000 lines of code were written to manage the various algorithms, data extraction, and visualisation tools. One of the biggest challenges was ensuring that the system could handle large datasets efficiently.

5.1 Performance Optimisations

As the number of CWEs and SbD strategies increased, the visualisation tool's performance began to degrade. Several optimisation techniques were implemented to improve responsiveness:

- **Lazy Loading:** Only render nodes and edges that are visible in the current viewport.
- **Zoom and Pan Optimisations:** Implemented dynamic zooming and panning, ensuring that only relevant parts of the graph are processed.
- **Optimised Search Functionality:** Integrated FlexSearch, an efficient full-text search engine that leverages contextual indexing to provide rapid search results. This allowed for real-time, highly performant queries on the dataset, ensuring users can quickly locate relevant vulnerabilities and mitigations.

These optimisations resulted in a significant performance improvement, allowing the tool to scale to hundreds of nodes without major slowdowns.

6 Results and Observations

After implementing the weighted set cover solution and visualising the relationships, the following results were observed:

- The tool successfully minimised the number of SbD strategies while covering all CWEs and properly associating them with their given SPARTA IDs.
- Visualising large datasets presented performance challenges, which were mitigated through optimisation techniques such as zooming and lazy loading of data.

7 Conclusion

This report demonstrated the successful integration of weighted set cover optimisation with a dynamic visualisation tool. The mapping between SPARTA IDs, CWEs, and SbD strategies was effectively modelled, and the tool provided meaningful insights into mitigating vulnerabilities.

A Source Code

The source code for this project, including the algorithm implementations and visualisation tools along with instructions on running a local copy, are available at: `https://github.com/Reantric/SPARTA`.

References

[1] SPARTA Framework Documentation. https://sparta.aerospace.org/.

[2] CWE Vulnerabilities Database. https://cwe.mitre.org/.

[3] Minimum Requirements for Space System Cybersecurity - Ensuring Cyber Access to Space. https://www.researchgate.net/publication/382524612_Minimum_Requirements_for_Space_System_Cybersecurity_-Ensuring_Cyber_Access_to_Space.

[4] Greedy Set-Cover Algorithms. https://www.cs.ucr.edu/~neal/Young08SetCover.pdf.