Visualizing Comparisons of Bills of Materials

Rebecca Jones*   Lucas Tate†

Pacific Northwest National Laboratory

ABSTRACT

The complexity of distributed manufacturing and software development coupled with the increasing prevalence of cyber and supply chain attacks necessitates a greater understanding of the hardware and software components that comprise equipment in critical infrastructure. When a vulnerability in a single software library can have disastrous consequences, being able to identify where that library may exist in equipment or software becomes a prerequisite for protecting the overall infrastructure. This need has sparked a large effort around the development and incorporation of bill-of-materials (BOM) into security, asset management, and procurement practices to aid in mitigating, and responding to future attacks. While much of the current research is devoted to creating BOMs, it is equally important to develop methods for comparing them to answer questions, such as: How has my software changed? Are two pieces of equipment equivalent? Does this piece of equipment that just arrived match my historical information? In this work, we demonstrate how BOMs can be represented by graph structures. We then describe how these structures can be fed into a graph comparison algorithm to produce a novel interactive visualization that allows us to not only identify differences in BOMs but show exactly where they are in the product.

Index Terms: Security and Privacy—Formal Methods and Theory of Security—Security Requirements; Human-centered computing—Visualization—Visualization Techniques—Graph Drawings

1 INTRODUCTION

Protecting critical infrastructure from cyber attacks, natural disasters, and other disruptions is a priority of the U.S. Government. Critical infrastructure includes providing electricity to homes and businesses, supplying natural gas for heating, and producing renewable energy sources. A loss of these services, as seen in the Solarwinds supply chain attack in 2020 [40], Texas snowstorm of 2021 [28], and the Colonial Pipeline cyber incident of 2021 [29]. In May 2021, the President of the United States issued an executive order to improve the country's cyber security [42]. As part of that order, every piece of software sold to the U.S. government must be accompanied by a software bill of materials (SBOM). A BOM is a detailed list of the components in the system and can describe hardware, software, operations, and Software as a Service (SAAS). The information in the BOM can be used to identify obsolete software as well as highlight potential susceptibility to publicly reported vulnerabilities [12]. Due to the mandate, industry has been exploring the generation of BOMs for their products.

The construction of BOMs today remains an inexact science for numerous reasons [45]. Some of that variation results from a lack of standardization. A primary reason for this is that there are currently competing formats and standards. BOMs also vary greatly depending on whether they were produced by a first-party such as

---

*e-mail: rebecca.d.jones@pnnl.gov
†e-mail: lucas.tate@pnnl.gov

the author/manufacturer with complete knowledge or by a third-party with incomplete knowledge. A current lack of mature tooling also increases the difficulty of reliably reproducing BOMs, particularly when looking at hardware BOMs which are often constructed manually. Recorded names or strings can vary widely due to convention, transcription, or spelling errors. Other differences can arise based on varying levels of completeness or depth (was every integrated circuit and stop accounted for, or every resistor soldered to the board recorded?). Beyond hardware or software components, the relationships linking them together can also be defined in a variety of ways. Relationships can be implied by a nesting structure, described explicitly, represented by a diagram, or possibly even omitted altogether.

Variation can also describe actual differences in composition, and that is exactly what BOMs are designed to capture. These differences could be alternative components that were used because they were cheaper, or even a component that had to be replaced because it's been operational for 15 years. Other differences might describe variations across a family of products or even the presence of counterfeiting. While comparing the competing standards is out of the scope of this paper, the inherent variability in BOMs necessitates tools that allow us to perform comparisons. The focus of our research is to provide an interactive visual comparison that effectively communicates how two BOMs may be similar or dissimilar to provide valuable insight and help to narrow subsequent analysis.

Current BOM comparison methods include using Excel or proprietary software such as Oracle Apps[1], Unisoft[2], ERPNext[3]. These tools are limited in the types of BOMs they accept and the data displayed, which does not necessarily include visualizations. Often, set comparisons are used, which lose the information of how the hardware or software is connected. They also focus on evaluating the differences between BOM versions and not necessarily distinctly different BOMs.

To account for the relationships between objects, we convert a BOM into a graph, which we can then easily compare and visualize. Traditional graph comparisons focus primarily on the structure of the graph but fail to take advantage of other information available within a BOM. To compare BOMs accurately, we need a method that allows us to incorporate important component information such as names, hashes, or versions, as well as structural information describing how those components fit together. We create a mapping that describes how the objects/components in one graph map to the components/objects in the other graph based on a depth-first search algorithm. When constructing the mapping, we can choose which information we want to consider (e.g. name, hash, name and hash) as well as whether the mapping should utilize exact or fuzzy matches. Fuzzy matching can be useful in instances where names or strings might have spelling or transcription errors, and can suggest where nodes in the graphs might have intended to reference the same component. Once constructred, the mapping is then used to combine the BOM graphs into a single merged graph.

1.1 Outline

In this paper, we will demonstrate our method for comparing BOMs and show how visualizing this approach can help analyze BOM differences. We will start out by discussing BOMs and how they can be represented as graphs, as well as current research in Sect. 2. Sect. 3 contains the method for combining BOMs, and Sect. 4 demonstrates the visualizations in two different examples. Lastly, we'll end with our conclusion in Sect. 5.

2 BACKGROUND

A bill of materials (BOM) is a list of all parts needed to produce a product. For each part in the finished product, the BOM can store various attributes like version, manufacturer, vendor, hash, package URL (PURL), and location. At this time, there is no single industry standard for generating BOMs. SBOMs have received more attention and presently, the three most widely used open source standards are CycloneDX [30], Software package data exchange (SPDX) [41], and Software identification (SWID) tags [11]. In addition to including information on the components, some BOMs also contain information on how the components are connected. This can be a separate document like a drawing or a list of relationships. The relationships in a BOM are important for identifying vulnerabilities, since connections may exacerbate or nullify a vulnerability.

2.1 Bill of Materials as Graphs

Many of the BOM comparison tools listed in Sect. 1 use set comparisons on the BOM objects. This results in a loss of information about how the components in the BOM are connected and is why relationships are needed in a BOM. For instance, a 12-pin port appearing in two hardware BOMs could be connected to different circuit boards in the actual products. There is also no guarantee that objects in BOMs have the same names, unique ids, or other metadata. This emphasis on needing connections naturally leads to representing BOMs as graphs.
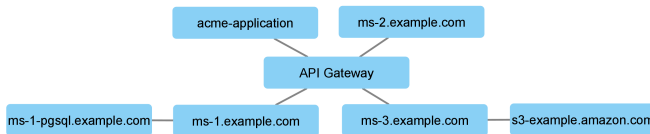

Figure 1: Visualization of an open-source Saas BOM. Created using Cytoscape [34].

A graph is a mathematical object that represents how things are connected. Each component of the BOM becomes a node in the graph while the relationships determine the edges. Edges are created between components when a component possesses some relationship (e.g. contains, uses, imports, includes, etc.) to another component. For example, if a software file imports another software file, then there would be an edge from one file node to the other file node. A hardware example is when a chip is soldered onto a circuit board, so there is an edge from the circuit board node to the chip node in the graph. Fig. 1 shows the graph for an open-source example SAAS BOM stored in a json file [1]. Each of the nodes in the graph represent one of the seven services listed in the BOM, uniquely identified by the "bom-ref". The edges are created from the dependencies key. Each entry in contains a "ref" and a "depends on" key which list the "bom-ref" ids that are connected.

Notice that since there could be duplicated parts in a system, there could be similar components in a BOM, e.g. multiple chips of the same type. In a BOM, each component has a unique ID, and different relationships to distinguish between the different components. Also, the metadata included in the BOM could be added to the graph as

node attributes, features that are attached to a node. For instance, each node could have a vender attribute that lists the product vendor.

2.2 Previous Research

As mentioned, a graph is a natural choice to represent a BOM that contains relationships [8, 18, 33, 43]. Hypergraphs can also be used to represent certain BOMs [27], although we note that not all BOMs have the structure needed to created a hypergraph. Graph databases are relational databases that store node data and relationships, i.e., graphs, instead of tables and have been suggested for storing BOM data [7]. A high profile commercial graph database, Neo4j, is used by the Army to manage their BOMs [44] quickly and efficiently [17]. In comparing BOMs, previous research has attempted to answer the question, "How similar are two BOMs?" This has often been answered with a distance metric [23, 32, 35]. There is some work that looks at combining multiple BOMs into graphs: aggregating multiple BOMs into one network to calculate network properties on the combined graph [9], and demonstrating graph matching methods to match pairs of BOMs to reduce production time [22].

There is more work on comparing general graphs. Methods for this include calculating distance metrics which determine how far apart graphs are in some metric space [4], clustering which looks at structural differences [6], deep learning [15, 19], and node correspondence algorithms [37]. We use the latter, in which nodes from one graph are mapped to another graph. This can be done where the node mapping is already known such as a cut distance [25] or where the node mapping is not known. When the node mapping is unknown, it is referred to as an unknown node correspondence, or UNC. One UNC approach is called alignment, where the mapping between graphs is created by optimizing a similarity function over the graph [24]. Many of the unknown node correspondence algorithms only use the graph structure and some are restricted to trees [26, 31]. The node mapping can also include the node features (in our case BOM metadata) in the similarity score calculation [13].

Visualizing graphs, especially large graphs, is challenging. The superposition approach overlays the two graphs [16]. This is similar to our method, in which we merge the two graphs into one and display the combined graph. Because there is more node metadata than can reasonably be presented at one time, a useful visualization for analysis will be interactive and allow users to manually compare the graphs [2, 20, 21]. Most BOM visualization techniques leverage tables to display the BOM differences [14, 36]. Often, these are compared using set comparisons, which leave out information about relationships, but they can facilitate multiple comparisons at once [3]. Visualizing sets can incorporate structure, but it's not as easy to see the differences [39]. SVG is a visualization tool that uses the Hungarian Assignment algorithm to map nodes and can include semantics like node attributes [5]. It requires node similarities to be calculated beforehand while our focus is calculating the mapping.

3 COMBINING TWO BOM GRAPHS

Our method of combining BOMs uses an unknown node mapping, where each node in one graph is mapped to a unique node in the second graph, if possible. To do this, we implement a depth-first search algorithm [38]. Depth first search is a traversal algorithm, meaning it visits each node in the graph exactly once and runs in linear time with the number of nodes in the graph. At each node, a similarity matching is done against the nodes in the other graph. The matching can be done on any number of node attributes and can be any string similarity metric, such as exact matching or Jaro-Winkler [10].

The process is outlined in Fig. 2. First, two BOMs are turned into graphs, as illustrated in Fig. 1. A starting node in Graph 1 is chosen that has a known match in Graph 2, such as C. Graph 1 is traversed, mapping nodes in Graph 1 to nodes in Graph 2 that have similar edges and attributes (or names as in the example). Then the graphs

are combined, using the node mapping. This yields the first merged graph in Fig. 2. Throughout the paper, blue represents nodes that have a mapping from Graph 1 to Graph 2, yellow represents nodes in Graph 1, and pink represents nodes only in Graph 2.

For further analysis, we include an option to predict nodes that might be the same based on fuzzy string matching and graph neighbors, as seen in the far right circle of Fig. 2. This is done by rerunning the depth first search algorithm with any fuzzy string matching algorithm on the nodes that were not mapped, D, F, and G in our example. Assuming H and D are close through some string similarity, since they have the same neighbors, a green edge is shown between them, depicting a possible match. For graphs where large number of nodes result in severe overplotting, we include an option to condense the merged graph by combining leaf nodes that appear in both graphs into a single supernode, as with nodes A and B.
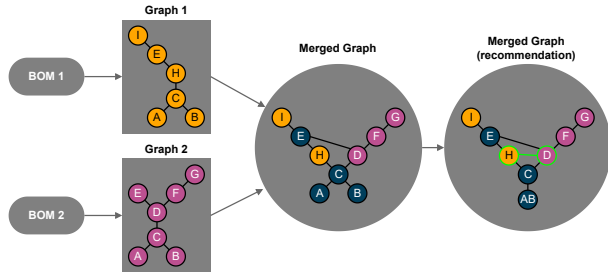

Figure 2: The process of combining BOMs into a single graph. Blue nodes indicate a successful mapping between the graphs. Pink nodes indicate that the node is only in Graph 1. Yellow nodes mean that the nodes appear in Graph 2. Green edges indicate that the connected nodes might be the same; they have similar neighbors and have node attributes that meet some similarity threshold.

An interactive .html version of the combined graph is saved, as well as a .gml, so the merged graph can be imported into most graph visualization tools. Our method is an end-to-end system, taking in graphs generated from BOMs, merging the graphs, and creating a visualization. Some of the benefits of our method are that we allow any type of graph (we're not restricted to trees), that we include the node attributes and structure in the node mapping, we allow for fuzzy string matching, and we include interactive visualizations.

4 USE CASES

To demonstrate the usefulness of this method, two illustrative use cases are provided. The first use case looks at two versions (1.6.3 and 1.8.0) of an open-source example SBOM for proton-bridge. The graphs, displayed in Fig. 3 and Fig. 4, were created in the same manner as the example in Sect. 2.1 where components become nodes in the graph and the relationships are created from the dependencies. The node mapping was done with an exact match on the SHA-256 hash. So nodes are considered equal if they have the same hash and the same neighbors. By looking at the merged graph shown in Fig. 5, we can easily see the differences between the two versions through the pink (version 1.6.3) and the yellow (version 1.8.0) nodes. When a user hovers over a node, its edges turn green to highlight the node's neighbors, and the node attributes are listed at the top. This way, a user can manually select and compare node information to identify discrepancies and possible differences.

From a cyber perspective, only the yellow nodes, nodes in the newer software version, will drive new behaviors or introduce new vulnerabilities, which should allow us to narrow our focus when analyzing new BOMs. The utility of comparing within a version controlled repository is diminished since much of the insights can be reproduced using version control software, so it's important to note that this approach is going to be agnostic to varied node definitions

(e.g. files, directories, functions, containers, etc.) or relationship definitions (e.g. contains, imports, installs, downloads, etc.) and furthermore allows the user to choose which node attribute(s) should be utilized for the mapping (e.g. names, hashes, etc.).
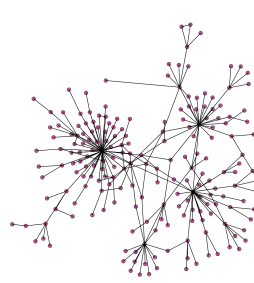

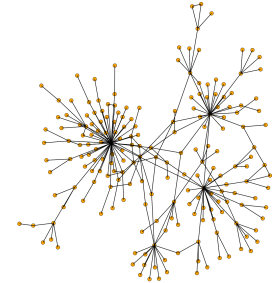Figure 3: Visualization of proton-bridge-v1.6.3 BOM.

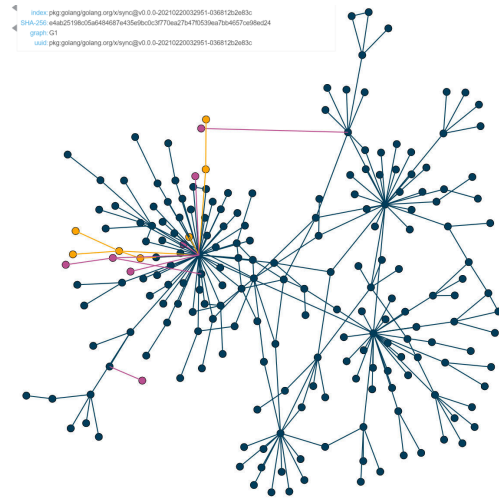Figure 4: Visualization of proton-bridge-v1.8.0 BOM.

Figure 5: Visualization of the combined proton-bridge graph. Blue nodes represent components with the same SHA-256 hash and edges. Pink nodes appear only in version 1.6.3, and yellow nodes only appear in version 1.8.0. The green edge is reflects relationships to the highlighted node. Node information for the selected node is displayed in the top left corner.

Next, we look at an actual comparison of two hardware bill of materials (HBOMs) Fig. 10. In this real example, the HBOMs were furnished by the same entity and generated for two devices of the same model, represented by Fig. 6 and Fig. 7. Nodes represent hardware components in the device and relationships show how the components are physically connected. A first pass of the algorithm used exact matching on the node names to generate the initial node mapping, creating the merged graph shown in Fig. 8. Blue nodes once again represent components in both BOMs. This means that the components have the exact same name and neighbors in both devices. We condensed the merged graph by collapsing blue leaf nodes into supernodes, as seen in Fig. 9. In a small graph like this one, collapsing the nodes is not needed, but when the graph becomes too large to easily display, collapsing leaf nodes can simplify the graph and allow the differences to be easily seen. A second pass of the algorithm used a fuzzy match on the names to draw green edges, see Fig. 10, which suggests where components may be similar.

At first glance we can see that there is a central blue structure in the graph which suggests the core structures of the two devices are the same. This isn't especially surprising since we are comparing two devices of the same model. From there we can visually make some additional observations. First, there is a yellow structure jutting from the center to the lower left. This suggests (and was confirmed) to be an additional circuit board in one of the devices accompanied by the corresponding mounted components.
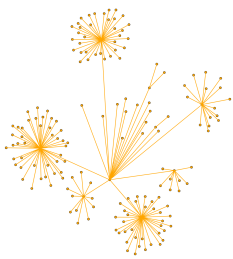

Figure 6: First HBOM Graph   Figure 7: Second HBOM Graph

Figure 8: Comparison of the exact component names between HBOMs generated on two distinct hardware devices of the same model. Blue nodes indicate mapped nodes in both graphs.

Second, the green edges point to instances of similarly named components at the same structural location in each device. By manually inspecting the nodes connected by green edges, subject matter experts (SMEs) were able to view the metadata and compare the two nodes. This quickly revealed similarly named components, spelling errors, and subtle differences in the convention used to record names. For example, $AS298$ mapped to $A5298$, likely a transcription error switching the S to a 5. The denser green webs form where there are multiples of a component on one device that got mapped to multiples of a component on the other device and the names are very similar.

The interactive visualization showed marked benefits over the baseline approach of set comparisons and greatly increased the ability of SMEs to explore, explain, and respond appropriately to discrepancies that were identified. Along with discovering the additional circuit board, SMEs were able to identify some major inconsistencies, likely spelling or transcription errors as described above, that had previously been undetected using this tool. With subsequent use

of the visualization SMEs were quickly able to form an intuition about the differences in similar graphs, greatly increasing the speed and efficiency of the BOM comparisons.

5 CONCLUSION

The increasing prevalence of BOMs requires new tools and methods to leverage them. While policy is largely driving the creation and adoption of BOMs, methods for analyzing them have lagged. Our method describes a flexible approach to easily explore the similarity or dissimilarity of two BOMs based on any number of metadata attributes. By presenting the results in an interactive graph visualization, users are able to quickly identify and explore differences for asset management or security use cases.
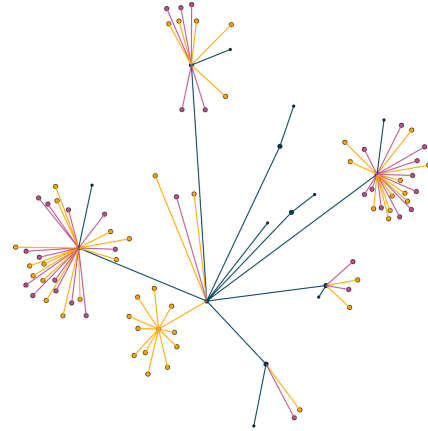

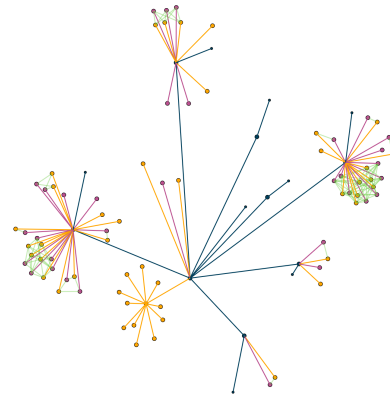Figure 9: Combined HBOM graph with leaf nodes collapsed.

Figure 10: Combined HBOM graph with suggested mappings shown in green.

REFERENCES

[1] CycloneDX/bom-examples. https://github.com/CycloneDX/bom-examples. Last accessed 2023-06-30, 2013.

[2] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 483–492. ACM. doi: 10.1145/2470654. 2470724

[3] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state-of-the-art of set visualization: The state-of-the-art of set visualization. 35(1):234–260. doi: 10.1111/cgf.12722

[4] N. Amenta and J. Klingner. Case study: visualizing sets of evolutionary trees. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pp. 71–74. IEEE Comput. Soc. doi: 10.1109/INFVIS.2002. 1173150

[5] K. Andrews, M. Wohlfahrt, and G. Wurzinger. Visual graph comparison. In *2009 13th International Conference Information Visualisation*, pp. 62–67. IEEE. doi: 10.1109/IV.2009.108

[6] D. Archambault. Structural differences between two graphs through hierarchies. In *GI '09: Proceedings of Graphics Interface 2009*, pp. 87–94. Canadian Information Processing Society.

[7] A. O. Aydin and A. Güngör *. Effective relational database approach to represent bills-of-materials. 43(6):1143–1170. doi: 10.1080/00207540512331336528

[8] S. Carmody, A. Coravos, G. Fahs, A. Hatch, J. Medina, B. Woods, and J. Corman. Building resilient medical technology supply chains with a software bill of materials. 4(1):34. doi: 10.1038/s41746-021-00403-w

[9] M. Cinelli, G. Ferraro, A. Iovanella, G. Lucci, and M. M. Schiraldi. A network perspective on the visualization and analysis of bill of materials. 9:184797901773263. doi: 10.1177/1847979017732638

[10] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. *IIWeb*, 2003, June 2003.

[11] Computer Security Division, Information Technology Laboratory. Software identification (SWID) tagging | CSRC | CSRC. https://csrc.nist.gov/projects/Software-Identification-SWID. Last accessed 2023-06-30.

[12] Cybersecurity and Infrastructure Security Agency. Software bill of materials (SBOM). https://www.cisa.gov/sbom. Last accessed 2023-06-30.

[13] J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence*, p. 323–327, August 2004.

[14] M. Freire, C. Plaisant, B. Shneiderman, and J. Golbeck. ManyNets: an interface for multiple network analysis and visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 213–222. ACM. doi: 10.1145/1753326.1753358

[15] T. Fujiwara, J. Zhao, F. Chen, and K.-L. Ma. A visual analytics framework for contrastive network analysis. In *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 48–59. IEEE. doi: 10.1109/VAST50239.2020.00010

[16] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. 10(4):289–309. doi: 10.1177/1473871611416549

[17] J. Guia, V. Gonçalves Soares, and J. Bernardino. Graph databases: Neo4j analysis:. In *Proceedings of the 19th International Conference on Enterprise Information Systems*, pp. 351–356. SCITEPRESS - Science and Technology Publications. doi: 10.5220/0006356003510356

[18] J. Guoli, G. Daxin, and F. Tsui. Analysis and implementation of the BOM of a tree-type structure in MRPII. 139(1):535–538. doi: 10. 1016/S0924-0136(03)00520-X

[19] D. Han, J. Pan, C. Xie, X. Zhao, and W. Chen. A visual analytics approach for structural differences among transportation networks. 53(5):566–571. doi: 10.1016/j.ifacol.2021.04.226

[20] M. Hascoët and P. Dragicevic. Interactive graph matching and visual comparison of graphs and clustered graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '12*, p. 522. ACM Press. doi: 10.1145/2254556.2254654

[21] J. Huerta-Cepas, J. Dopazo, and T. Gabaldón. ETE: a python environment for tree exploration. 11(1):24. doi: 10.1186/1471-2105-11-24

[22] M. Kashkoush and H. ElMaraghy. Matching bills of materials using tree reconciliation. 7:169–174. doi: 10.1016/j.procir.2013.05.029

[23] M. Kashkoush and H. ElMaraghy. Product family formation by matching bill-of-materials trees. 12:1–13. doi: 10.1016/j.cirpj.2015.09.004

[24] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj. Topological network alignment uncovers biological function and phylogeny. 7(50):1341–1354. doi: 10.1098/rsif.2010.0063

[25] Q. Liu, Z. Dong, and E. Wang. Cut based method for comparing complex networks. 8(1):5134. doi: 10.1038/s41598-018-21532-5

[26] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. 22(3):453–462. doi: 10.1145/882262.882291

[27] L. Nagy, T. Ruppert, A. Löcklin, and J. Abonyi. Hypergraph-based analysis and design of intelligent collaborative manufacturing space. 65:88–103. doi: 10.1016/j.jmsy.2022.08.001

[28] National Weather Service. Valentine's week winter outbreak 2021: Snow, ice, & record cold. https://www.weather.gov/hgx/2021ValentineStorm. Last accessed 2023-06-30.

[29] Office of Cybersecurity, Energy Security and Emergency Response. Colonial pipeline cyber incident. https://www.energy.gov/ceser/colonial-pipeline-cyber-incident. Last accessed 2023-06-30.

[30] OWASP Foundation. OWASP CycloneDX software bill of materials (SBOM) standard. https://cyclonedx.org/. Last accessed 2023-06-30.

[31] A. Priel and B. Tamir. A vectorial tree distance measure. 12(1):5256. doi: 10.1038/s41598-022-08360-4

[32] C. Romanowski and R. Nagi. On comparing bills of materials: A similarity/ distance measure for unordered trees. 35(2):249–260. doi: 10.1109/TSMCA.2005.843395

[33] M. Schmidt, B. Gehring, J.-S. Gerber, J. M. Stocker, M. Kreimeyer, and M. Lienkamp. Graph-based similarity analysis of BOM data to identify unnecessary inner product variance. In *21st International Conference on Engineering Design (ICED17)*, vol. 1, pp. 489–498, August 2017.

[34] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. 13(11):2498–2504. doi: 10.1101/gr.1239303

[35] H. M. Shih. Product structure (BOM)-based product similarity measures using orthogonal procrustes approach. 61(3):608–628. doi: 10. 1016/j.cie.2011.04.016

[36] O. Shilovitsky. OpenBOM best practices – BOM visualization types. https://www.openbom.com/blog/openbom-best-practices-bom-visualization-types. Last accessed 2023-06-30, September 2022.

[37] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi. Comparing methods for comparing networks. 9(1):17557. doi: 10.1038/s41598-019-53708-y

[38] R. Tarjan. Depth-first search and linear graph algorithms. 1(2):146–160. doi: 10.1137/0201010

[39] A. Telea and D. Auber. Code flows: Visualizing structural evolution of source code. 27(3):831–838. doi: 10.1111/j.1467-8659.2008.01214.x

[40] D. Temple-Raston. A 'worst nightmare' cyberattack: The untold story of the SolarWinds hack. https://www.npr.org/2021/04/16/985439655/a-worst-nightmare-cyberattack-the-untold-story-of-the-solarwinds-hack. Last accessed 2023-06-30.

[41] The Linux Foundation Projects. International open standard (ISO/IEC 5962:2021) - software package data exchange (SPDX). https://spdx.dev/. Last accessed 2023-06-30.

[42] The White House. Executive order on improving the nation's cybersecurity. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/. Last accessed 2023-06-30.

[43] Y. Wang, W. G. Wang, and S. Mao. A new type of BOM model and its application. 347-350:1234–1238. doi: 10.4028/www.scientific. net/AMM.347-350.1234

[44] J. Webber. Top 10 use cases: Bill of materials. https://neo4j.com/blog/top-10-use-cases-bill-of-materials/. Last accessed 2023-06-30.

[45] G. Wright and K. Hannah. S4 SBOM challenge debrief. https://www.youtube.com/watch?v=2sQNcc4xphw. S4x23.