Interactive Visualization and Text Mining For the CAPEC Cyber Attack Catalog

Steven Noel
The MITRE Corporation
7515 Colshire Drive
McLean, Virginia, USA
snoel@mitre.org

ABSTRACT
The Common Attack Pattern Enumeration and Classification (CAPEC™) is a publicly-available enumeration and classification of cyber-attack patterns. CAPEC provides a catalog of known cyber attacks, along with standard language for describing attack classes and their hierarchical relationships. The CAPEC catalog is a complex hierarchy of hundreds of attack classes, representing textual, categorical, and referential data. We describe a number of analytic and visual techniques for interacting with this comprehensive knowledge base. We visualize the CAPEC attack pattern hierarchy, which shows taxonomic relationships of attack classes and sub-classes. We also visualize hierarchical clusters based on the textual content of CAPEC entries, providing an automated and mathematically sound alternative to the current ad hoc manual process. For interacting with referential data, we employ bipartite graph visualizations of attack-pattern references to standardized software weakness.

Author Keywords
Attack patterns; Common Attack Pattern Enumeration and Classification (CAPEC); graph visualization; hierarchical clustering; dendrograms.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces.

INTRODUCTION
Cyber security requires taking the attacker's perspective, to understand the approaches for exploiting software systems. On the other hand, it is time consuming to gain a deep understanding of potential attacks and to express them in a sufficiently general way. There is strong motivation for a standardized way of expressing such knowledge.

This role is served by the *Common Attack Pattern Enumeration and Classification* (CAPEC™) [15], a standardized catalog of common attack patterns. CAPEC characterizes individual attack patterns, and also defines parent-child relationships for organizing attack patterns into a hierarchy from general to specific.

Navigation of CAPEC web content relies on following parent-child hyperlinks in textual content. This does not lend well to understanding the overall hierarchical structure of CAPEC. To address this, we introduce new web-based interactive visualization techniques for CAPEC attack-pattern classes.

While CAPEC defines a taxonomy of attack-pattern classes, this is defined by CAPEC developers, through an ad hoc process. On the other hand, the CAPEC content itself provides a rich source for clustering related classes and retrieving relevant information. We apply a vector-space approach, modeling CAPEC attack patterns as document term vectors. We define distances that measure document vector similarity. We then perform cluster analysis to discover groups of related attack patterns, e.g., for expanding potential patterns of interest.

CAPEC is part of *Making Security Measurable* [6], a collection of initiatives for information sharing in cyber security. CAPEC references other related standards, such as the Common Weakness Enumeration (CWE™) for software weaknesses and Common Vulnerabilities and Exposures (CVE®) for known vulnerabilities. We describe interactive bipartite graph visualizations for navigating these complex many-to-many referential relationships.

PREVIOUS WORK
Graph/tree visualization has been well studied [4][17]. The graph visualization problem is ill-posed; there are many possible and potentially conflicting criteria to characterize visual quality, and certain aspects are NP-complete [5]. The usual approach is to design heuristics that optimize particular aesthetics. In practice, such heuristics work especially well for the case of trees, as we apply here.

The web interface for navigating the CAPEC taxonomy (at https://capec.mitre.org/) has the common "tree view" design, as shown in Figure 1. This design is limited in terms of depicting the complex hierarchic relationships of CAPEC as a whole. To provide this overall view, we cast the CAPEC

partially-ordered (acyclic) parent-child graph to a corresponding tree, then apply various interactive tree visualizations.

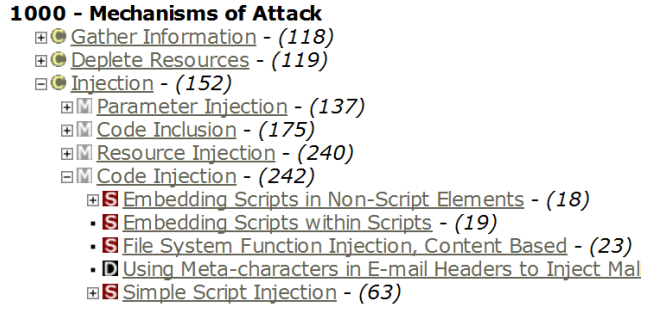

Figure 1. CAPEC tree view interface.

We employ a vector-space approach [19] to model CAPEC as a document collection. We then apply vector operations for measuring document similarity. We perform hierarchical clustering, visualized as dendrograms [12][16]. This provides an automated and mathematically sound alternative to manual ad hoc processes. Our interactive visualization interfaces (including trees and bipartite graphs) are implemented in D3.js [2]. This represents the first application of such visualization techniques to CAPEC.

VISUALIZING THE CAPEC TAXONOMY

Figure 2 shows the graph of parent-child relationships in CAPEC (for the default view CAPEC-1000, *Mechanism of Attack*). Each vertex is an attack pattern or category. An edge points from parent to child. The graph is 459 vertices (14 categories and 445 patterns) and 511 edges.

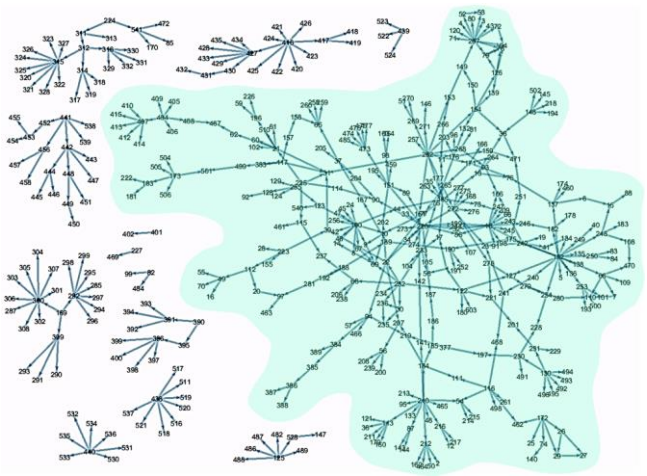

Figure 2. Full CAPEC parent-child taxonomy graph.

In Figure 2, the CAPEC taxonomy graph contains a number (weakly) connected components: twelve relatively small trees, and the larger (shaded) component. The shaded component is acyclic, i.e., a partial ordering of parents to children. The presence of only a few non-tree elements are enough to visually mask the hierarchical structure of the shaded part of the figure. In fact, of the 459 vertices in the CAPEC graph, 395 of them (86%) have a single (or no) parent as required for a tree.

To transform the partially-ordered (shaded) portion of the CAPEC graph to a corresponding forest, we find the full hierarchy of children beneath each root (non-child) node. The result is 30 distinct hierarchies, where a hierarchy is a root vertex and its full sub-graph of descendants. Among the 30 CAPEC hierarchies that we construct, 25 are rooted trees. The remaining 5 hierarchies are acyclic graphs (partial orders) with a single root. Through the relatively small degree of node replication, we preserve all existing edges (parent-child relationships) of the original graph.

For tree visualization, a Cartesian layout (such as Figure 1) is not the most efficient use of display space. One way to pack a tree display into a smaller space is to employ a radial layout [10]. Also, there is a class of visualization techniques that represents tree relationships implicitly [11], rather than explicitly drawing vertices and edges.

One such technique is the sunburst visualization [14], which draws radial wedges for each level of the tree. Figure 3 is a sunburst visualization for our CAPEC taxonomic tree. If we click on a wedge, it zooms to the selected sub-tree.

There is a class of tree visualization techniques designed to fill the display space completely, known as treemaps [5]. For rectangular treemaps, it is sometimes difficult to distinguish among levels for larger trees. Also, aspect ratios can be very different for rectangles of equal area, making visual comparisons of size more difficult. An alternative is circular treemaps [18], which uses nested circles instead of rectangles. While circular treemaps do not fill space completely, their nesting structure is easy to recognize. Also, the aspect ratio is the same (a circle) for all sub-trees, making size comparisons more obvious.

Figure 4 shows the circular treemap for CAPEC. This visualization gives larger display areas for higher levels of the tree, thus emphasizing the more general (encompassing) attack patterns. This differs from the sunburst visualization (Figure 3), whose radial design has a bias for lower levels, because wedges further from the center have more area. The top of Figure 4 is the full CAPEC tree. If we click on a circle, it zooms in to that sub-tree (bottom of the figure).

Another approach is treemaps that recursively divide the display space via Voronoi tessellation [1]. These have near-unity aspect ratio and stable interactive dynamics [13]. Figure 5 is a Voronoi treemap visualization [3] for the CAPEC parent-child tree. Here, the cells at each level are weighted by their number of children (attack patterns), so that patterns with more children have more display area. The top of the figure shows the full CAPEC tree. If we click on a cell, it zooms in to show that sub-tree.

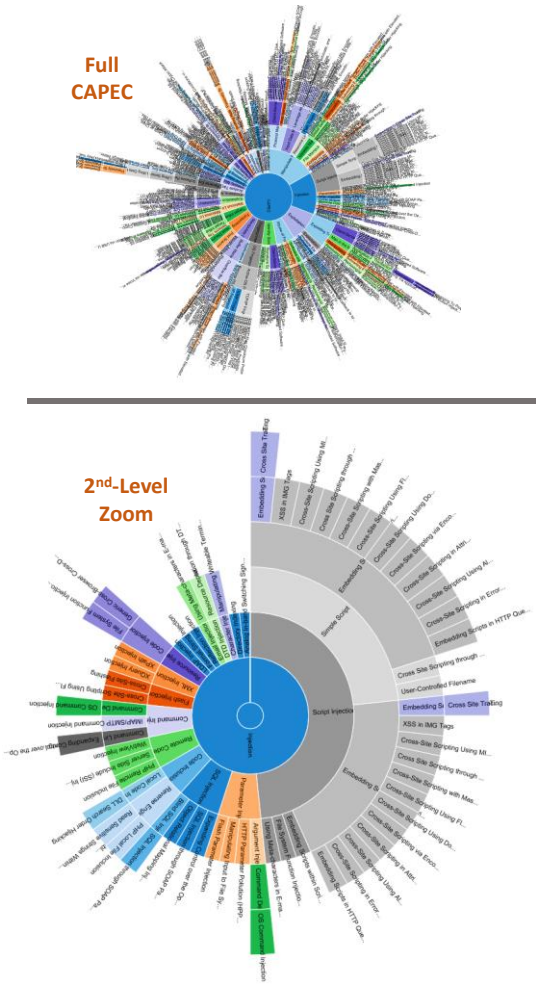

Figure 3. Sunburst visualization of CAPEC tree.

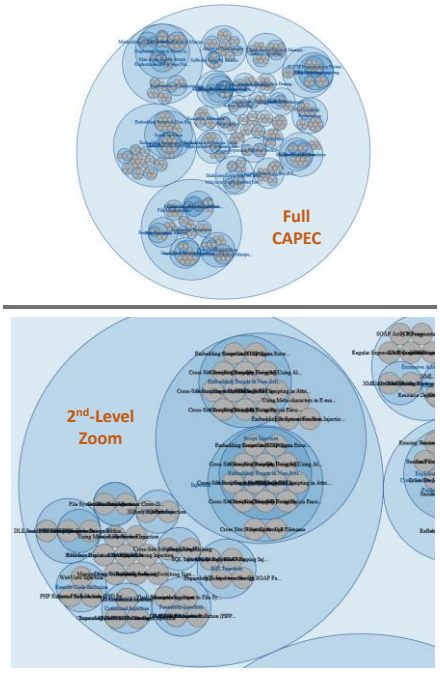

Figure 4. Circular treemap of CAPEC tree.

Figure 5. Voronoi treemap of CAPEC tree.

MINING CAPEC TEXT

Much of the CAPEC content is natural-language text. While natural language provides complete freedom of expressiveness, it introduces challenges for analysis of attack patterns.

We treat each CAPEC attack pattern as a document containing text to be mined. We apply a vector-space model in which each document (attack pattern) is a vector whose components represent the terms (relevant words) in the document. From this representation, we define distances from documents to query terms, as well as inter-document distances. For information retrieval, we use query-document distances to rank documents by their strength of query match. We use document distances to cluster documents for exploratory analysis, i.e., discover groups of related attack patterns based on their language content. We do this through hierarchical clustering, showing clusters at ranges of distances.

Vector-Space Model

In the vector-space model for text mining [19], documents are represented as vectors, with components representing document terms. The weight of a component indicates the strength (e.g., frequency) of the term in the document. We represent each CAPEC attack pattern as a document vector.

Across the entire text corpus of selected attributes (name, summary, and prerequisites) over all CAPEC patterns, there

are 4313 unique words. Figure 6 shows the frequency of these words within the corpus. This distribution approximately follows "Zipf's law" [20], in which term (word) frequency is inversely proportional to term rank. On a double logarithmic scale as in Figure 6, such a power-law distribution appears as a straight line with negative unity slope. In a power-law distribution, a term of rank k occurs $1/k^{th}$ as often as the most frequently occurring word. Thus relatively few common words comprise much of the corpus, and there are many infrequent words.

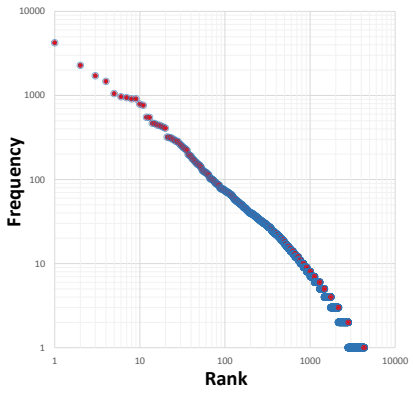

Figure 6. Frequency distribution of CAPEC words.

In selecting words to serve as terms in our document vectors, the usual approach is to exclude such frequently occurring non-content-bearing words (so-called "stop words"). We apply a threshold of 30 stop words for our CAPEC analysis.

In vector-space models, the terms for a document vector are given weights, denoting the relevant importance of each term within the document. Various weightings have been proposed, each having advantages and disadvantages in terms of efficiency and performance (e.g., for search results), often depending on characteristics of the text corpus [9]. These are often some variation of "term frequency-inverse document frequency (tf-idf)," in which a document term weight is proportional to its frequency within the document, and inversely proportional to its frequency over the entire corpus (more common words overall have reduced weight).

Attack Pattern Distances

We can also apply our vector-space distances to measure distances among documents themselves, to cluster related documents. This allows us to discover groups of related CAPEC attack patterns based on their language content. We apply the standard tf-idf weighting [8] to the document vectors. We have document vector $\boldsymbol{d}_j = (d_{1,j}, d_{2,j}, \cdots, d_{t,j})$, where component $d_{i,j}$ represents term i in document j. The weight $w_{j,i}$ for term i is then

$$w_{i,j} = f_{i,j} \cdot \log \frac{|D|}{|\{d \in D | i \in d\}|}$$

Here, $f_{i,j}$ is the frequency of term i in document j, $|D|$ is the total number of corpus documents, and $|\{d \in D | i \in d\}|$ is the number of documents that contain term i.

This composite tf-idf weighting is composed of factors to enhance mutual recall and precision of terms when comparing documents. The term frequency $f_{i,j}$ enhances recall, in the sense that a document's frequently-occurring terms should help in recalling it. On the other hand, terms that occur frequently in a large documents can lead erroneous retrieval, so that precision suffers. The inverse document frequency $\log(|D|/\{d \in D | i \in d\})$ addresses that by reducing weights for terms that occur frequently throughout all documents.

Given this weighted-term vector representation, we can compute the distance between a pair of documents. For this, we measure the angle (in high-dimensional term space) between the document vectors. Documents with a small angle between them are deemed more similar. We can leverage the identity $\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \cos\theta$, where θ is the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$. We then define cosine similarity as

$$similarity_{cos}(\boldsymbol{a}, \boldsymbol{b}) \equiv \cos\theta = (\boldsymbol{a} \cdot \boldsymbol{b})/(|\boldsymbol{a}||\boldsymbol{b}|)$$

Certain analyses such as clustering assume distances (dissimilarity), i.e., the complement of similarity. A proper distance metric (obeying the triangle inequality) requires the similarity to be expressed as an angle and normalized:

$$similarity_{\theta}(\boldsymbol{a}, \boldsymbol{b}) \equiv 1 - \frac{2\cos^{-1}(similarity_{cos})}{\pi}$$

The document distance is then simply the complement:

$$distance(\boldsymbol{a}, \boldsymbol{b}) = 1 - similarity_{\theta}(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\cos^{-1}(similarity_{cos})}{\pi}$$

Figure 7 shows the resulting distance matrix for the first 50 CAPEC attack patterns. Here, matrix element $distance(\boldsymbol{d}_i, \boldsymbol{d}_j)$ is the distance between documents (CAPEC attack patterns) $\boldsymbol{d}_i$ and $\boldsymbol{d}_j$. For plotting, self-distances (main diagonal) are at the distance midpoint (versus their actual value of zero) to reduce dynamic range.

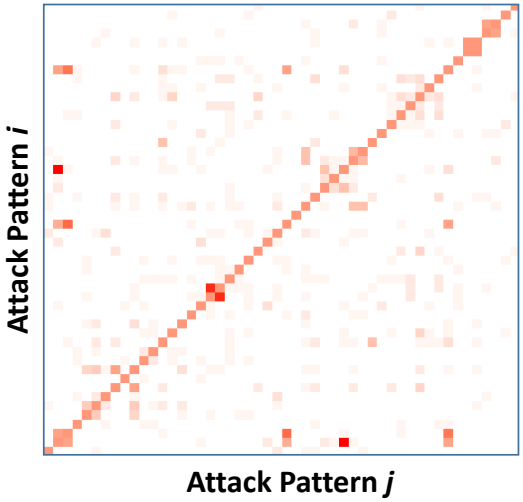

Figure 7. Distances for first 50 CAPEC attack patterns.

Attack Pattern Clustering

Given distances between CAPEC attack patterns, we can cluster them into groups. For exploratory analysis, hierarchical clustering is helpful. This allows us to see the tradeoff between numbers of clusters and cluster size (number of members), and the threshold distances at which the clusters form.

We can visualize hierarchical clustering as a dendrogram. This is a tree visualization, with the leaves of the tree representing the documents (CAPEC attack patterns). The top of the diagram shows the clustering threshold distance. The dendrogram shows the tree of cluster composition at each possible level of distance (clustering) threshold.

Figure 8 shows the complete-linkage dendrogram for the first 50 CAPEC attack patterns (distances in Figure 7). The selected threshold distance forms 8 clusters. The two closest attack patterns (distance of 0.65) are CAPEC-10 *Buffer Overflow via Environment Variables* and CAPEC-13 *Subverting Environment Variable Values*. These attack patterns are siblings in the CAPEC hierarchy of attack patterns, i.e., they are specializations of parent pattern (CAPEC-77 *Manipulating User-Controlled Variables*). The next-closest pair of attack patterns in Figure 8 are CAPEC-114 *Authentication Abuse* and CAPEC-115 *Authentication Bypass*. CAPEC-114 and CAPEC-115 are also siblings in the CAPEC hierarchy (children of CAPEC-225 *Exploitation of Authentication*).

Thus far, our text mining has been applied to these CAPEC attributes: attack pattern name, summary, and attack prerequisites. The resulting clusters are therefore based on general characteristics of attack patterns, along with conditions for their success. However, there are a variety of other CAPEC attributes, which characterize attack patterns in different ways. Mining these other attributes can tell us how attack patterns are related in terms of these other characteristics.

As an example, consider Figure 9. This is the complete-linkage dendrogram for the outcomes of each detailed attack execution step of the attack patterns. In this case, we mine the CAPEC content for commonalities in impact on compromised cyber assets. This dendrogram has 79 attack patterns, which are all those having defined attack step outcomes in CAPEC. We select only those outcomes marked as 'Success,' to characterize attack patterns in terms of successful outcomes for the attacker (cyber asset impact).

In Figure 9, these two pairs of attack patterns are the closest:

- CAPEC-63 *Simple Script Injection* and CAPEC-199 *Cross-Site Scripting Using Alternate Syntax*
- CAPEC-32 *Embedding Scripts in HTTP Query Strings* and CAPEC-86 *Embedding Script (XSS) in HTTP Headers*

The distances for each of these pairs (CAPEC-63/199 and 32/86) are near zero. The pairs then merge into a cluster of four at a distance of about 0.2. Careful examination of these four attack patterns shows that the language (for attack outcomes) is nearly identical for all. The key difference is that CAPEC-63 and CAPEC-199 have additional outcomes:

- A list of application user interface entry fields is created by the attacker.
- A list of resources accessed by the application is created by the attacker.

This additional content causes the 0.2 difference in distances between CAPEC-63/199 and 32/86 in the dendrogram.

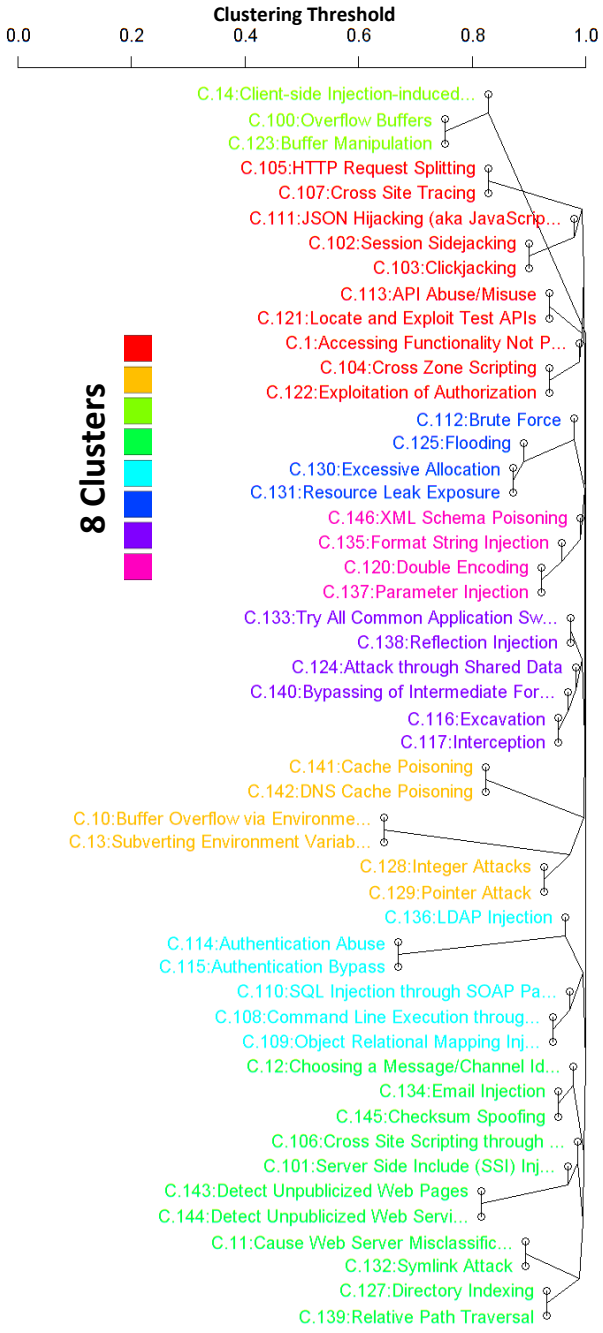

Figure 8. Dendrogram for first 50 attack patterns.

The closest attack patterns in this dendrogram are part of a cluster of eight CAPEC entries (shaded in Figure 11):

1. CAPEC-18 *Embedding Scripts in Non-Script Elements*
2. CAPEC-19 *Embedding Scripts within Scripts*
3. CAPEC-32 *Embedding Scripts in HTTP Query Strings*
4. CAPEC-63 *Simple Script Injection*
5. CAPEC-86 *Embedding Script (XSS) in HTTP Headers*
6. CAPEC-91 *XSS in IMG Tags*
7. CAPEC-199 *Cross-Site Scripting Using Alternate Syntax*
8. CAPEC-244 *Cross-Site Scripting via URI Schemes*

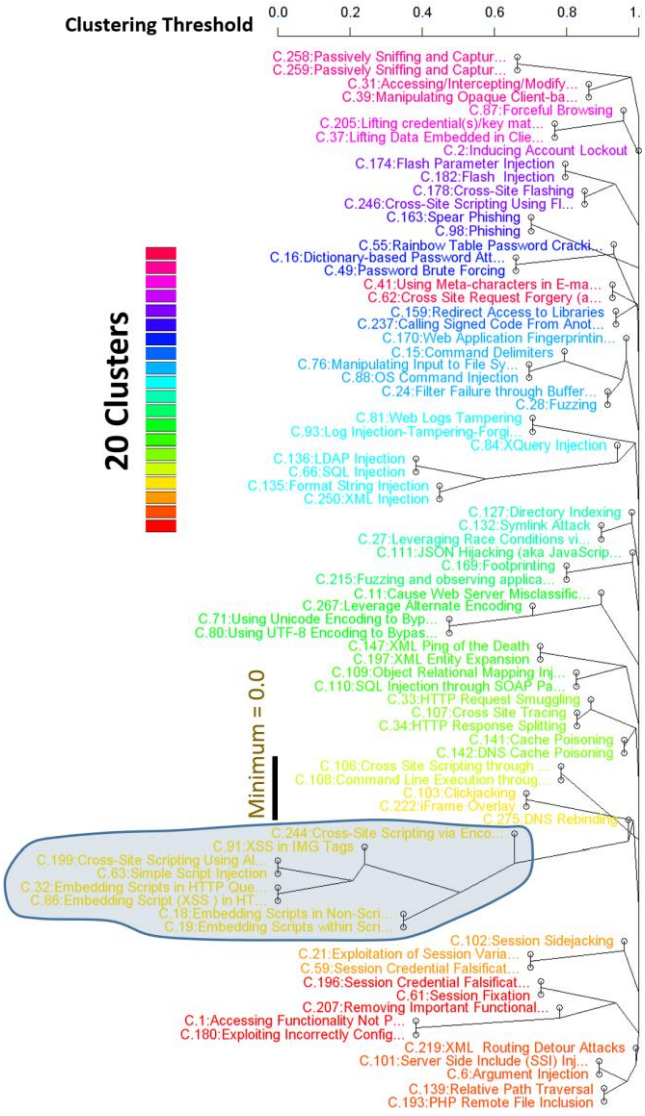

Figure 9. Dendrogram for attack execution outcomes.

These attack patterns are within a distance of 0.65; the next closest pattern to this cluster is almost one away. As shown in Figure 10, these attack patterns are also nearby in the CAPEC classification hierarchy. Other nearby attack patterns that are not part of the cluster of eight (e.g., CAPEC-73, 209, 247, 245, 243, and 198) are missing definitions for attack-step outcomes. This is because CAPEC is a work in progress, and not all content is defined for all attack patterns.

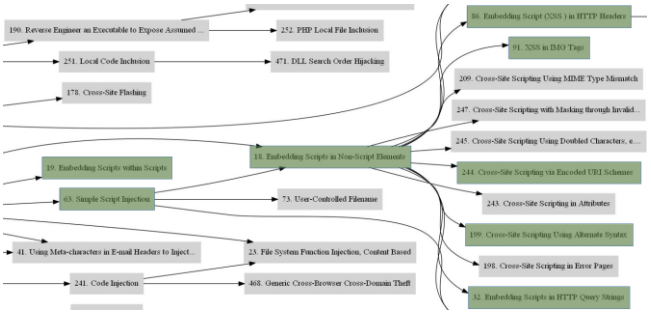

Figure 10. Cluster of similar attack patterns within context of CAPEC hierarchy.

CAPEC places attack patterns into categories according to particular (subjective) criteria. We apply our clustering to categories to explore similarity structure at the highest level of the CAPEC hierarchy. Figure 11 shows the resulting complete-linkage clustering dendrogram, using terms from category name and summary.

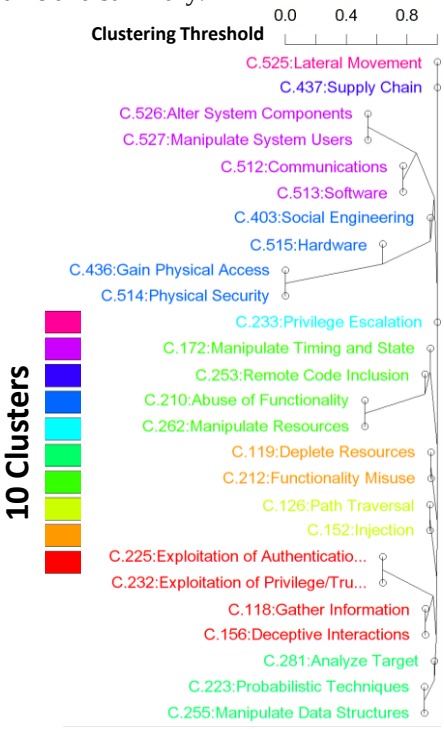

Figure 11. Dendrogram for attack pattern categories.

The dendrograms in Figure 8, Figure 9, and Figure 11 are each based on distances computed from different CAPEC content. Figure 8 is derived from general characteristics of attack patterns and the conditions for their success, Figure 9 is based on attack execution outcomes, and Figure 11 is from general descriptions of attack pattern categories.

Figure 12 shows the distribution of document distances for each of these types of CAPEC content. The distribution for attack pattern name, summary, and prerequisites (top curve, corresponding to Figure 8) has a sharper knee. This indicates that most documents are far away from one another, with relatively few close documents (attack patterns). In fact, the

closest distance for all pairs is quite high (0.53). On the other hand, the distribution for attack pattern outcomes (middle curve, corresponding to Figure 9) has a gentler knee (more attack patterns that are similar to one another), and a closest distance of zero.

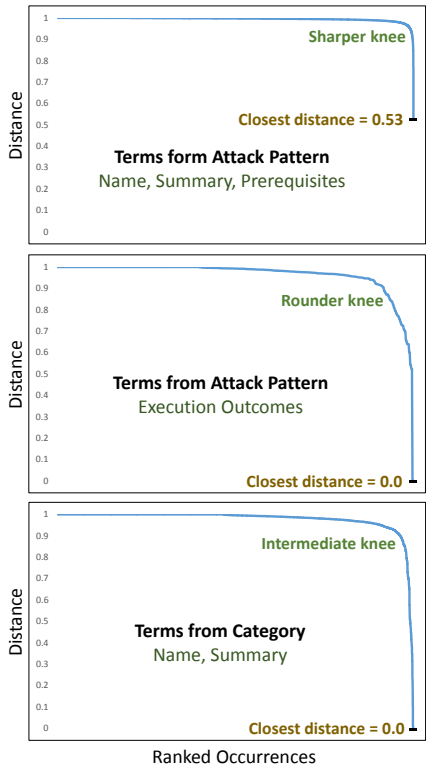

Figure 12. Distances for different CAPEC content.

For attack pattern categories (bottom curve, corresponding to Figure 11), the knee of the distribution is intermediate in sharpness. This suggests that the text content for attack pattern categories (name and summary) is more discriminatory than for the general attack pattern attributes (name, summary, and prerequisites), and less so than for attack step outcomes of attack patterns.

VISUALIZING CAPEC REFERENTIAL DATA

CAPEC includes references to other standards under Making Security Measurable, e.g., CWE (software weaknesses) and CVE (system vulnerabilities). Often, these cross references need to be examined as a whole, e.g., for trending or coverage. Or, we might need to know which attacks are applicable to weaknesses or vulnerabilities in our enterprise, or specific weaknesses that an attack exploits. These cross references are many-to-many: each CAPEC attack pattern can reference multiple CWEs and/or CVEs, and each CWE or CVE can be referenced by multiple attack patterns. Visualization can help us understand these complex relationships, as shown in Figure 13.

Figure 13 shows the software weaknesses (CWE identifiers) referenced by a set of CAPEC attack patterns. The left column is CAPEC attack patterns that reference CWE weakness in the right column. The references are shown as (bipartite) graph edges between the CAPECs and CWEs. The counts on the left are the numbers of CWEs that each CAPEC references; the counts on the right are the numbers of CAPECs that reference each CWE.

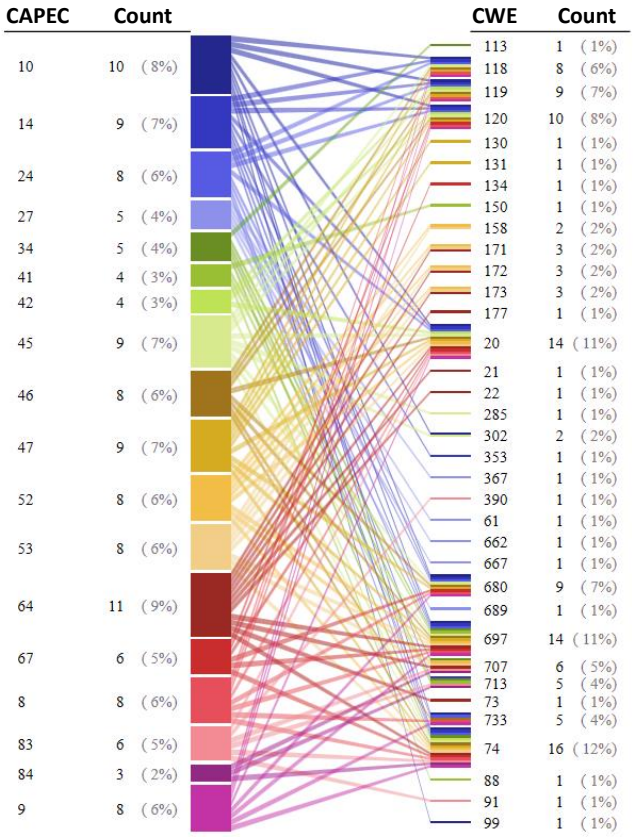

Figure 13. CWE weaknesses for selected attack patterns.

These 18 attack patterns in Figure 13 are chosen to meet a specific set of attack scenario criteria [7]. This includes target architectural paradigm, attack purpose, technical impact, severity, and likelihood. Because of having to meet multiple criteria, these attack patterns are alike in many ways. Still, as Figure 13 shows, these attack patterns exploit a variety of different software weaknesses.

We can focus on specific mappings by hovering over them in the visualization (in either the CAPEC or CWE column). On the left side of Figure 13, we hover over CAPEC-10 (*Buffer Overflow via Environment Variables*), and on the right side we hover over CAPEC-27 (*Leveraging Race Conditions via Symbolic Links*). We see that these two attack patterns reference two distinct sets of weaknesses.

While these two attack patterns have similar values for several key attributes, their attack mechanisms are quite different. CAPEC-10 leverages environmental variables to overflow memory buffers, while CAPEC-27 races the system to create a symbolic link to malicious content. These differences in mechanism of attack are reflected in the disjoint sets of weaknesses for these two attack patterns.

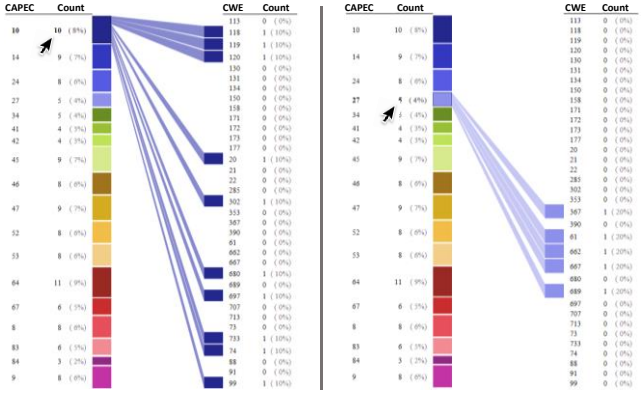

Figure 14. Weaknesses for individual attack patterns.

CONCLUSION

We describe a variety of web-based interactive visualization techniques for the CAPEC catalog of attack patterns. This includes visualizing the overall hierarchical structure of CAPEC attack patterns, organized from general to specific.

We also apply text mining to the natural-language content of CAPEC. We compute hierarchical clusters and visualize them through dendrograms. In this way, we group related attack patterns through automated analysis, versus human experts grouping them by subjective criteria. We also describe bipartite graph visualization for navigating complex collections of cross references from CAPEC attack patterns to standardized CWE software weaknesses.

Our visualization and analytic techniques provide a range of new capabilities for understanding and interacting with the rich content and relationships in CAPEC. For example, CAPEC content builders can refine hierarchical structures to better match linguistic similarities, discover outliers, etc. Our analyses can also help for building higher-level security models such as attack graphs.

ACKNOWLEDGMENTS

The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed.

REFERENCES

1. Balzer M. and Deussen O. Voronoi Treemaps. *Proc. IEEE Symposium on Information Visualization 2005*.
2. Bostock, M., Ogievetsky, V., and Heer, J. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011).
3. Carrot Search. FoamTree: Interactive Voronoi Treemap. http://carrotsearch.com/foamtree-overview.
4. Graham, M. and Kennedy, J. A Survey of Multiple Tree Visualization. *Information Visualization 9*, 4 (2010).
5. Johnson, B. and Shneiderman, B. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. *Proc. IEEE Conference on Visualization 1991*.
6. Marriott, K. and Stuckey, P. NP-Completeness of Minimal Width Unordered Tree Layout. *Journal of Graph Algorithms and Applications 8*, 3 (2004).
7. Martin, R. Making Security Measurable and Manageable. *Crosstalk – The Journal of Defense Software Engineering* (2009). 26-32.
8. Noel, S. *Leveraging the Common Attack Pattern Enumeration and Classification (CAPEC™) for Attack Modeling*. MITRE technical report WN140076 (2014).
9. Robertson, S. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. *Journal of Documentation 60*, 5 (2004). 503-520.
10. Salton, G. and Buckley, C. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 32*, 4 (1996). 431-443.
11. Schulz, H. Treevis.net: A Tree Visualization Reference. *IEEE Computer Graphics and Applications*. 31, 6 (2011). 11-15.
12. Schulz, H.-J., Hadlak, S., and Schumann, H. The Design Space of Implicit Hierarchy Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics 17*, 4 (2011). 393-411.
13. Seo, J. *Information Visualization Design for Multidimensional Data: Integrating the Rank-by-Feature Framework with Hierarchical Clustering*. Dissertation, University of Maryland, College Park (2005).
14. Sud, A., Fisher, D., Lee, H.-P. Fast Dynamic Voronoi Treemaps. *Proc. International Symposium on Voronoi Diagrams in Science and Engineering 2010*. 85-94.
15. The Common Attack Pattern Enumeration and Classification. https://capec.mitre.org/
16. Venables, W. and Ripley, B. *Modern Applied Statistics with S-Plus*. Springer-Verlag (1994).
17. von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.-D., and Fellner, D. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum 30*, 6 (2011). 1719-1749.
18. Pebbles – Using Circular Treemaps to Visualize Disk Usage. http://lip.sourceforge.net/ctreemap.html.
19. Wong, S., Ziarko, Raghavan, V., and Wong, P. On Modeling of Information Retrieval Concepts in Vector Spaces. *ACM Transactions on Database Systems 12*, 2 (1987). 299–321.
20. Zipf, G. *Human Behavior and the Principle of Least Effort*. Addison-Wesley (1949).