# Hierarchical Incident Clustering for Security Operation Centers

David Silva
Symantec Research Labs
david_silva@symantec.com

Matteo Dell'Amico
Symantec Research Labs
matteo_dellamico@symantec.com

Michael Hart
Symantec Corporation
michael_hart@symantec.com

Kevin A. Roundy
Symantec Research Labs
kevin_roundy@symantec.com

Daniel Kats
Symantec Research Labs
daniel_kats@symantec.com

## ABSTRACT

In security operation centers (SOCs) of large organizations, triaging and remediating security incidents is a tedious and error-prone job. Incidents are generated by correlating security-related events that may indicate attack (e.g., unblocked end-point alerts on the same machine). In many cases, several incidents consist mainly of the same sets of events, and they can be traced back to the same root cause. Rather than requiring analysts to triage incidents individually or scan incident lists one by one to identify related incidents, in our demo we provide hierarchical clustering of incidents so that analysts can quickly identify all similar incidents and perform joint remediation actions. The hierarchical aspect of the clustering algorithm enables analysts to navigate a cluster tree, allowing them to find the right granularity at which incidents should be grouped so that a single remediation action can dispatch the largest possible number of related incidents. Our user interface simultaneously provides insight into the nature of attacks on an organization, and is also an efficient mechanism to automate response. *This is a demo paper. Our demo can be viewed at https://vimeo.com/250453752 with password* `svp3267b`. *Should this paper be accepted we will make the demo video public.*

**ACM Reference format:**
David Silva, Matteo Dell'Amico, Michael Hart, Kevin A. Roundy, and Daniel Kats. 2018. Hierarchical Incident Clustering for Security Operation Centers. In *Proceedings of KDD 2018 Workshop on Interactive Data Exploration and Analytics, London, England, August 20, 2018 (IDEA'18)*, 6 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Security Operation Centers (SOCs) often use tools that aggregate *events* generated by a variety of security devices and products, then use rule-based engines to generate *incidents*, which are collections of these events happening on the same machine and around the same time (e.g., on the same day). An illustration of this can be seen in Figure 1. These incidents can vary in severity and degree of confidence whether something is actually an attack (for example, incidents can be triggered by anomalous network traffic patterns that prove to be benign). Much of a SOC analyst's time is spent on the tedious task of triaging incidents, and either responding to them individually, or manually recognizing sets of similar incidents to be collectively acted upon [16].

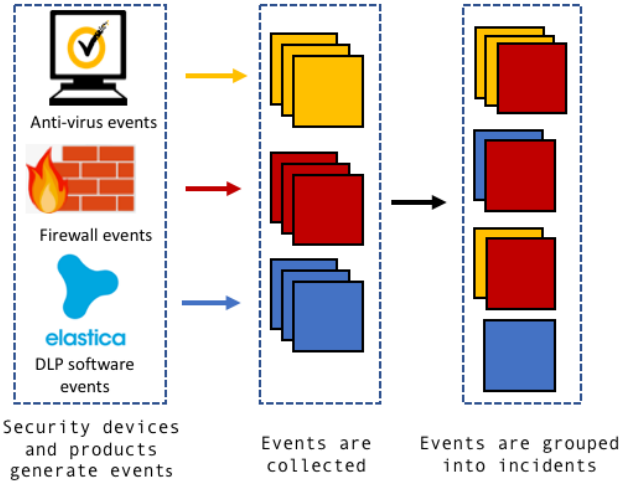


**Figure 1: Events are generated by security products, are aggregated, and then grouped into incidents based on their proximity to each other in space and time**

It is not surprising that spending time on these menial tasks affects the analysts' productivity and happiness [15]; for this reason we propose a tool that enables analysts to quickly dispatch sets of related incidents with joint remediation action. The ultimate goal is to enable analysts to spend more time on rewarding and intellectually stimulating tasks such as investigating the root causes behind the security incidents and securing the company's systems to prevent recurrences of similar problems.

We provide an interactive tool that takes as input a set of incidents, and outputs a *hierarchical clustering*. This enables analysts to:

- quickly discover similar new incidents and act on them together;
- associate new incidents with similar reviewed ones, and find out how they were handled in the past;
- navigate the hierarchical structure to find the right cluster granularity, such that collective actions can be taken on sufficiently similar incident sets.


*IDEA'18, August 20, 2018, London, England*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

We make the following contributions:

(1) we provide a clustering algorithm that is both scalable and flexible, in that it allows for arbitrary distance functions on the features of the security incidents that it clusters;
(2) the hierarchical clustering allows incident data to be explored both from the top down, to help understand the "big picture" and composition of the incident pool, and from the bottom up, starting from highly similar incidents and gradually expanding the incident pool;
(3) our approach does not force every single incident into a cluster: unique incidents will remain displayed as isolated elements;
(4) our interactive user interface provides visual cues to illustrate properties of incident clusters, such as cluster size and homogeneity, incident severity and predominant incident category.

## 2 SYSTEM DESCRIPTION

Our system consists of two components. The first is the clustering component, which collects incident data and computes a hierarchical clustering on that data; the second one is an interactive user interface that allows users to explore and take action on the results of the clustering. We describe both in the following.

### 2.1 Clustering

Our choice for the clustering algorithm is dictated by a few requisites:

(1) the algorithm should be *flexible*, so we can define an *arbitrary distance function* between our data items, to effortlessly improve our results when the design of the distance function improves;
(2) it should be *hierarchical*, to enable interactive exploration of the dataset, with the option of "zooming" in and out between smaller and larger clusters;
(3) it should be *scalable* to work on large data sizes;
(4) it should be *effective* where the output represents patterns in the data.

The next section describes the design of the distance function. We then describe the algorithm used.

*2.1.1 Design of the Distance Function.* Each incident consists of a list of events happening on the same machine on a given day, as well as contextual metadata. The clustering algorithm considers each incident $i$ as a vector of events $E_i$, where $E_{i,k}$ is the number of times the event whose identifier is $k$ is represented in incident $i$. We adopt a sparse feature representation because most incidents contain few non-zero event counts. That is, we represent the vector $E_i$ as a dictionary of values in which we create entries $E_{i,k} = v$ when $v \neq 0$ and not otherwise, so that all undefined entries of $E_{i,k}$ indicate that $v = 0$ for event $k$ and incident $i$. The additional metadata is not used to cluster data, but it is shown to users in the interface described in Section 2.2.

Events vary widely in terms of their severity and rarity. While all events are collected as potentially relevant to security, some of them represent common events (e.g., a user logged in, a user entered an incorrect password), which are relevant only when happening many times. Since the most common events are generally also the least relevant ones, we adopt a TF.IDF normalization [9], and work on vectors $N_i$ such that $N_{i,k} = v$ if event $k$ has $v$ times the average number of instances across all incidents:

$$N_{i,k} = n \frac{E_{i,k}}{\sum_{k'} E_{i,k'}}.$$

Like the $E_i$ vectors, we represent the sparse $N_i$ vectors using dictionaries.

We finally use the generalized Jaccard distance as our distance measure between the $E_i$ vectors:

$$d(i,j) = 1 - \frac{\sum_k \min\left(N_{i,k}, N_{j,k}\right)}{\sum_k \max\left(N_{i,k}, N_{j,k}\right)}.$$

*2.1.2 Clustering Algorithm.* Our clustering algorithm is inspired by NG-DBSCAN [10], a scalable and approximated implementation of the DBSCAN [4] density-based clustering algorithm that can be applied to arbitrary data and distance functions. NG-DBSCAN implements a variant of the NN-Descent [2] algorithm, and caches the distances computed between items to feed them as inputs to the DBSCAN algorithm. DBSCAN is well-known as an algorithm that provides good clustering and isolates "noise" (i.e., non-clustered items) from the actual clusters.

DBSCAN is not a hierarchical clustering algorithm, therefore with an approach analogous to NG-DBSCAN, we run NN-Descent to compute distances between item pairs, and feed these results to our implementation of the OPTICS [1] tool, which is a hierarchical generalization of DBSCAN.

OPTICS orders the data items and outputs a *reachability graph*, which places ordered items on the $x$ axis and their *reachability distance*, which is a measure of an item's distance from the previous item in the graph, on the $y$ axis.

In the reachability graph, data items are ordered so that similar items are close to each other. *Valleys* in this plot represent clusters, whose width represents the number of items they contain, and whose depth indicates the degree to which the cluster differs from other data items. OPTICS is a generalization of DBSCAN because the clusters of DBSCAN for a parameter $\varepsilon$ can be obtained by finding the consecutive points in the plot for which the reachability distance is lower than $\varepsilon$.

In Figure 2, we show the results of running our algorithm on a set of incidents in a single organization: we can see how the clusters vary in terms of both width and depth in the reachability plot. We can also see hierarchical clustering structures that appear in the plot as valleys within valleys. The coloring of the plot indicates the degree to which security analysts responded to the incidents in question: red indicates incidents with which responders interacted (e.g., by escalating them or by indicating that they took action to resolve the issue), while pale yellow indicates incidents that were ignored, and which remained in the queue until they timed out. It is clear that the amount of interaction is correlated with the clusters that we can observe; in particular, there is a large cluster of events which time out in most cases, while other clusters that consistently elicit a response. The largest cluster of incidents that is consistently ignored consists mostly of "Suspicious Traffic"

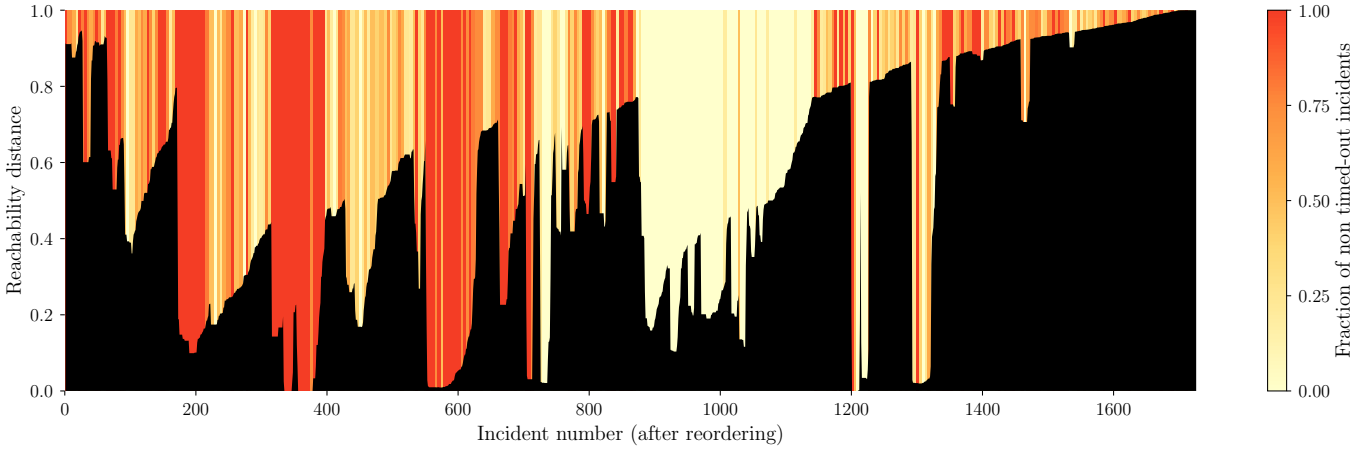


**Figure 2: The reachability plot output by OPTICS on a set of incidents for a single organization, along with the ratio of non timed-out incidents: those on which the analysts of a company took action.**

incidents. It is possible that responders did not know how to handle this vague category of incidents.

## 2.2 User Interface

The graphical user interface (GUI) explores the neighborhood of a given *current incident*. The GUI is sub-divided in two interdependent visual components: the hierarchical clustering Tree and the Incidents Grid, both shown in Figure 3 on the following page. Using the Tree, SOC analysts can navigate the hierarchical clusters generated by our algorithm. Selecting a cluster in the Tree causes the Incidents Grid to show the cluster's associated security incidents, and for these incidents to be examined at multiple levels of detail. We proceed by describing the Tree and Incident Grid and conclude this section by introducing the Action Bucket feature and a use-case scenario.

*2.2.1 The Clusters Tree.* The hierarchical clustering is represented by a tree, whose root contains all incidents in the queue. Each tree node represents a cluster of similar incidents, and its child nodes represent incident clusters that are subsets of the parent. A node's parent cluster is a larger cluster with lower similarity between incidents; conversely, a node's child clusters contain fewer, more similar, incidents. Not all nodes in a cluster are represented in its child clusters because singleton items that are not sufficiently similar to other incidents are not forced into any cluster.

To ease navigation, only clusters that contain the current incident and their siblings are displayed. The current cluster (displayed in the Incident Grid) is displayed with a yellow border, while all other clusters that contain the current incident have a blue border. Incidents have a type attribute which broadly categorizes them, such as "reported malware download" or "ransomware infection", and each node is labeled with two values: the number of incidents in the cluster and the number of distinct incident types in the cluster.

Each node is visualized through a geometric shape (by default, a circle). Up-pointing triangles (△) represent clusters where at least $p$% (by default, $p = 70$) incidents were responded to in the past by security analysts, while down-pointing triangles (▽) represent clusters of incidents having at least $p$% false positives in the past. Inner node colors inform about incident severity: red nodes represent clusters with at least $p$% high-severity incidents, while green nodes represent those with at least $p$% low-severity ones.

*2.2.2 The Incidents Grid.* When a cluster is selected in the Tree, the Incidents Grid shows a view with one row for each incident type in the cluster. Clicking on the lens symbol for an incident type opens a view of the individual incidents having that type, and a listing of the security events associated with that incident, sorted (by default) by the descending relevance of each incident (defined as the sum of the $N_{i,k}$ values for incidents $i$ in the cluster and event $k$). It is possible to perform additional sorting and filtering (e.g., only showing incidents that do or do not include a security event of type $k$).

*2.2.3 Action Buckets.* *Action buckets* are containers suited for incidents that should undergo a common resolution action. Users can create and label buckets, add incidents and associate actions to them. During cluster exploration, users can add incidents to buckets through the "update bucket" widget; at the end of the investigation, users can commit a common action to incidents in the same bucket. It is worthy to notice that committed buckets can, in the following, be regarded as a kind of ground truth for the clustering algorithm, and drive the development of new and better distance functions, improving our definition of Section 2.1.1. Custom filters can be associated to each bucket, in order to hide incidents and speed up the triaging process.

## 3 DEMO OUTLINE

We consider the use case where our user, a SOC analyst, is working on an incident queue. The investigation loop starts with the user selecting an incident and starting the investigation process. Let us suppose that the user starts the investigation with an incident having type *Trojan XYZ*; in the following we provide an example of the interaction.

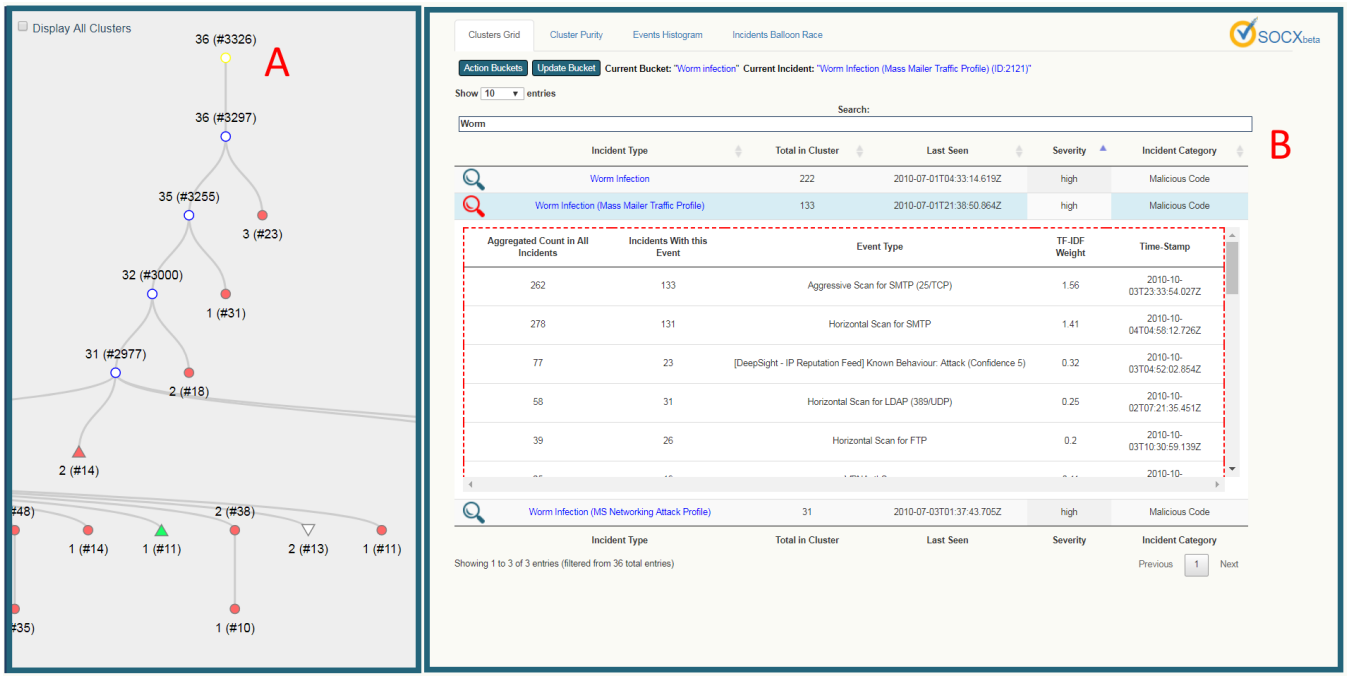


**Figure 3: The Core graphical user interface (GUI) is composed by the Hierarchical Clusters Tree and by the Incidents Grid, highlighted as components A and B respectively.**

(1) By engaging in a bottom-up approach, the user reaches the deepest level of the cluster hierarchy containing the current incident and verifies that this cluster contains 16 incidents having a single type. In this cluster, most items have high severity (node inner color is red), but no significant actions were taken in the past (node has a cicle shape).
(2) By clicking the node in the Tree, the contents of the cluster are displayed in the Grid. The user conveniently assesses the available contextual information (IP address, machine ID, sub-network, etc.) and creates a new action bucket, *Trojan XYZ for Network XPTO*, to stage relevant incidents.
(3) Following the bottom-up approach, the analyst turns their attention back to the Tree and looks at the current node's parent contains 23 instances of incidents that also trace back to the incident Trojan XYZ. Surprisingly the shape of the cluster is an upside-down triangle, meaning that a major percentage of the incidents within the cluster were in the past considered false positives. Now looking at the contextual data in the Grid, more machines from other internal sub-network are also being targeted with the same type of Trojan. As a consequence, the user creates a second action bucket to accommodate the new findings.
(4) The sibling of the previous node contains 10 more incidents of the same type which end up within a different cluster. By examining the Grid and the incident events, the user understands that some events differ, but the root cause is still the same; additionally, the contextual information confirms that these incidents share the same IP address of those found in step 2. Hence, the analyst adds the incidents to the action bucket created in step 2, which now contains a total of 26 incidents.
(5) The analyst can repeat the last steps and consult ancestors and siblings of the current nodes, to find even more related incidents. When this investigation is considered over, appropriate actions are associated to the event buckets and submitted; the user returns to the list of queued incidents and restarts an investigation loop.

## 3.1 Video

Our demo is shown in a video available at https://vimeo.com/250453752; the password needed to watch it is `svp3267b`. We will release a public version of the video with the camera-ready version of this paper.

# 4 VALUE AND CONTRIBUTION

We think that the valuable time of security experts should not be wasted on repetitive and tedious tasks. The tool was designed to enable analysts to spend more time on gratifying and productive endeavours.

The key characteristics of our clustering algorithm are:

**Flexibility.** We use a generic algorithm that clusters arbitrary data according to arbitrary distance functions. Our current approach adopts generalized Jaccard distance on TF.IDF-normalized vectors, but a large amount of additional meta-data is collected, and plenty of domain-specific knowledge could be encoded in the distance function. We plan to iterate on our distance function design by interacting with analysts, and encode their expertise in better distance functions; the

distance function is arbitrary (e.g., we don't require it to be a metric), which gives us the best possible flexibility for future improvements.

**Hierarchy.** Our data is naturally clustered in a hierarchical fashion: as we can see in Figure 2 on page 3 there are clusters within clusters of incidents. Our hierarchical visualization allows to navigate between a big picture and close-up views of the incident panorama, providing context and letting analysts choose the right level of granularity to operate on.

**Scalability.** Our algorithm inherits a computational complexity of around $O\left(n^{1.1}\right)$ from NN-Descent [2]. This enables us to cluster even large numbers of incidents, which is useful when using a database of historical incidents to provide context for the new incidents that analysts need to take action on.

**Density-Based Approach.** Many clustering algorithms "force" each data point into a cluster. This is not desirable in our case, since incidents that are not naturally part of an incident cluster should not be mixed with those that are indeed similar between each other. Density-based clustering algorithms, instead, recognize that some data points are isolated and do not put them in a particular cluster. This is of key importance in a context where rare events that are different from the typical workload can be a sign of advanced attacks carried out by skilled adversaries. Additionally, the output of density-based algorithms is *explainable*: one point is clustered together with others only if there are at least $k$ ($k$ is an algorithm parameter) other items that are close to it, and these close points can be shown to the user.

Our User Interface (UI) is the instrument of control for the underlying clustering process. It moves away from the standard queuing models where incidents are presented in a sequential fashion, and it provides a peripheral view of the threats landscape where users can select the exact level of granularity to act upon a given occurrence. The key characteristics are:

**Portability and Extensibility.** Our interface is portable because its underlying technologies are all based on JavaScript. Currently the prototype has been deployed as a Kibana Dashboard, a Splunk Application and a Web SPA (Single Page Application). Custom extra visualizations of the hierarchical clustering were manually crafted and are also available to better understand each clustering layer. Additionally the visual pieces involved in highlighting data-set properties can be easily customized. For example: defining which properties and thresholds are involved when describing a high-severity cluster.

**Usage Telemetry.** The analyst's navigation of the cluster tree and incidents grid are recorded to enable tuning of the clustering algorithm and user interface, and to assess whether our interface improves the consistency with which incidents are handled.

## 5 RELATED WORK

In this Section, we discuss related work in two areas: clustering for security-related events and visualization of hierarchical clustering.

*Clustering Security-Related Events.* Various pieces of work are devoted to the problem of clustering security-related data. We discuss some work that target the clustering of security events, but remark that our approach clusters *security incidents*, which consist of multiple security events of various types, whereas the prior work has focused on clusters of individual security events.

A recurring characteristic of much of the prior art is that it requires manual effort to annotate the data in various ways. Julisch and Dacier [6, 7] aggregate intrusion detection events according to a manually specified hierarchical taxonomy over the attributes of the events. Ning et al. [12] and Porras et al. [13] delegate even more control to the domain experts that analyze the data, wherein each cluster is formed according to a clustering constraint that is explicitly specified by the user. The approach of Mathew et al. [11] requires extensive hand-tuning and annotation (including modeling of possible attacks) to enable real-time analysis of security-related events as they unfold. Other approaches [17, 18] require that a distance function be defined for each attribute of the analyzed items, and a set of weights to allow fusing each of them.

Compared to the above pieces of work, we consider that our approach has two main benefits: *flexibility* and *hierarchical output*. Our clustering is flexible in that it allows a variety of custom distance functions to be applied to each attribute, ranging from the simple distance function described in Section 2.1.1 to distance functions that encode domain knowledge and are iteratively refined by experts. Our algorithm's hierarchical output is important because it allows experts to explore related incidents and the coarsest granularity at which multiple similar incidents can be jointly resolved.

*Visualization for Hierarchical Clustering.* A standard way to visualize hierarchical clusters is to use *dendrograms* [5], i.e., a taxonomic tree diagram with elements on the $x$ axis and distances on the $y$ axis: a split on the tree at height $y$ means that two clusters are split at distance $y$. In the most general case, the leaves of the dendrograms are individual data items; it is therefore not surprising that dendrograms of large datasets are difficult to read. Visualizations have accordingly been developed to represent large hierarchical clusters, such as treemaps [14], radial layout trees [3], and botanical trees [8]. While each of these and other methods have their own particular advantages, they all share the goal of densely representing large hierarchies and providing an understanding of the overall hierarchical structure of the data. Without minimizing the merits of these approaches, we chose to base our visualization of the clustering hierarchy as a simple tree structure because we require a visualization that is well suited to interactive exploration of the hierarchy both from the bottom up and from the top down, relying on the right pane of our visualization to provide an interactive summary of the clusters themselves.

## 6 CONCLUSION

Security Operations Centers report high levels of analyst burnout because the job promises to be creative and intellectually stimulating, but is instead dominated by tedious and repetitive investigations of similar security incidents [15]. We develop clustering and incident visualization tools to eliminate redundant effort. Rather than imposing a rigid clustering algorithm with a fixed granularity,

we provide the first hierarchical clustering tool for security incidents, thereby enabling analysts to quickly identify the context in which a security incident has occurred and analyze and dispatch large numbers of related security incidents at once.

## REFERENCES


[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. In *ACM Sigmod record*, Vol. 28. ACM, 49–60.

[2] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*. ACM, 577–586.

[3] Peter Eades. 1992. Drawing Free Trees. *Bulletin of the Institute of Combinatorics and its Applications* (1992), 10–36.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[5] Tal Galili. 2015. dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering. *Bioinformatics* 31, 22 (2015), 3718–3720. https://doi.org/10.1093/bioinformatics/btv428

[6] Klaus Julisch. 2003. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information Systems Security* 6 (2003), 443–471.

[7] Klaus Julisch and Marc Dacier. 2002. Mining intrusion detection alarms for actionable knowledge. In *KDD*.

[8] E. Kleiberg, H. van de Wetering, and J. J. van Wijk. 2001. Botanical visualization of huge hierarchies. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.* 87–94. https://doi.org/10.1109/INFVIS.2001.963285

[9] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

[10] Alessandro Lulli, Matteo Dell'Amico, Pietro Michiardi, and Laura Ricci. 2016. NG-DBSCAN: scalable density-based clustering for arbitrary data. *Proceedings of the VLDB Endowment* 10, 3 (2016), 157–168.

[11] Sunu Mathew, Daniel Britt, Richard Giomundo, Shambhu Upadhyaya, Moises Sudit, and Adam Stotz. 2005. Real-time multistage attack awareness through enhanced intrusion alert clustering. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*. IEEE, 1801–1806.

[12] Peng Ning, Yun Cui, and Douglas S Reeves. 2002. Analyzing intensive intrusion alerts via correlation. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 74–94.

[13] Phillip A Porras, Martin W Fong, and Alfonso Valdes. 2002. A mission-impact-based approach to INFOSEC alarm correlation. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 95–114.

[14] Ben Shneiderman. 1992. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics* 11 (1992), 92–99.

[15] Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Jacob Case, Xinming Ou, Michael Wesch, John McHugh, and S. Raj Rajagopalan. 2015. A Human Capital Model for Mitigating Security Analyst Burnout. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. USENIX Association, Ottawa, 347–359. https://www.usenix.org/conference/soups2015/proceedings/presentation/sundaramurthy

[16] Sathya Chandran Sundaramurthy, John McHugh, Xinming Ou, Michael Wesch, Alexandru G Bardas, and S Raj Rajagopalan. 2016. Turning Contradictions into Innovations or: How We Learned to Stop Whining and Improve Security Operations. In *Symposium on Usable Privacy and Security (SOUPS)*.

[17] Olivier Thonnard and Marc Dacier. 2008. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 154–163.

[18] Olivier Thonnard, Wim Mees, and Marc Dacier. 2010. On a multicriteria clustering approach for attack attribution. *ACM SIGKDD Explorations Newsletter* 12, 1 (2010), 11–20.