A Dataset of Networks of Computing Hosts

Omid Madani, Sai Ankith Averineni
Cisco Secure Workload, Palo Alto, CA 94301, USA
omadani|saverine@cisco.com

Shashidhar Gandham*
Meta, Menlo Park, CA 94025, USA
gshashi@fb.com

ABSTRACT

We are making public a dataset of 21 disjoint graphs representing communications among machines running different distributed applications in various enterprises. We provide a ground truth grouping for one graph. The grouping is useful for evaluating tasks such as clustering hosts based on network communications. We describe the graphs and present a brief exploratory analysis to illustrate some of the properties, possible uses of the data, and some of the challenges.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; • **Networks** → *Network monitoring*; *Network management*.

KEYWORDS

unsupervised learning, community detection, clustering, network communications, network security, graphs, datasets

ACM Reference Format:
Omid Madani, Sai Ankith Averineni and Shashidhar Gandham. 2022. A Dataset of Networks of Computing Hosts. In *Proceedings of the 2022 ACM International Workshop on Security and Privacy Analytics (IWSPA '22), April 24–27, 2022, Baltimore, MD, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3510548.3519368

1 INTRODUCTION

We are providing a dataset containing 21 disjoint graphs obtained from the deployment of our service, Cisco Secure Workload, by different customers representing diverse verticals (health, banking, government, ...). The graphs represent the communications among hosts running various distributed applications, recorded in spans of several consecutive hours, over two or more days.[1] For privacy reasons, we have de-identified the data: we have assigned integer ids to IPs, *i.e.* the nodes in the graphs, and changed the numbers of all the communication ports used. The dataset can be used for exploring network analysis techniques, such as community discovery, ranking nodes by similarity (such as similarity in neighborhood and ports used), and clustering hosts. These graphs were obtained by software sensors developed by the Tetration Analytics group of Cisco (now, Cisco Secure Workload). We provide a ground truth grouping of nodes, based on function, for one graph (Section 2.4).

In our use case, we are interested in the automated discovery and effective presentation of groupings of the nodes (hosts) and their communication patterns. We have developed tools that provide a user, such as a security administrator of an organization, visibility into their complex network, and help the user partition the network, often hierarchically, into groups corresponding to geographic areas, different departments, and hosts running different applications in the organization. This organizing and naming of the groups is a necessary step in defining and maintaining *network security policies*, aka network (micro) segmentation: hosts in different groups (segments) can only communicate on a few well-defined and restricted channels [6, 10, 15]. Such policy enforcement severely limits penetration and spread of malware and hackers. This step of grouping hosts and assigning meaningful names/labels to the groups, with the human in the loop, is also highly useful in generating insights, for example in uncovering broad patterns of communications and normalcy modeling, with applications not just to security but also for network troubleshooting and performance improvements. We note that in our use case, we provide the system and the associated set of network analysis tools to the customer, and the tools should function independent of us: we do not, in general, get to select or tune, or personalize, the algorithms based on customer data.

The dataset made available by Turcotte et al. [16] contains a rich set of network flow as well as host information (diverse host events) for their enterprise network. Our dataset is limited to the communications[2] (the graphs, no host events), but we expect that our dataset offers further diversity of different distributed workloads, which we have not found in other sources of network datasets [3, 4, 8]. Ground truth grouping for network data is rare [8] and expensive, and we hope that our groupings will be useful for validating graph algorithms such as graph-based clustering. We next describe the data and report on brief experiments that shed light on some properties of the graphs and possible uses.

2 DESCRIPTION OF THE DATA

An edge in any graph corresponds to a connection from a client (or consumer) node to a server (or provider) node. With each edge, we also report the server port and the protocol used, 6 for TCP and 17 for UDP. For a directed edge from node A to node B on port number 10, we say node A is a client of node B on service port 10 (and B provides 10). The server port designation is a best effort via several techniques[3] using the attributes of the protocol, the

*Work done while at Cisco Secure Workload.
[1]We plan to make the data available at the UCI machine learning repository [4].

IWSPA '22, April 24–27, 2022, Baltimore, MD, USA

ACM ISBN 978-1-4503-9230-3/22/04…$15.00
https://doi.org/10.1145/3510548.3519368

[2]We provide a subset of the fields they provide. For instance, we do not include the client port.
[3]The TCP handshake includes the information, but may be missed by the sensor, *e.g.* if the sensor does not see the start of flow (long flows) or due to limited sensor availability (*e.g.* imposed computational constraints). The UDP protocol does not require a handshake, and determination of which end is the server is can be more challenging in general than TCP.

G	$\|V\|$	$\|E_u\|$	d_m	d_{max}	ports	ports2	$\|E_d\|$	p_{1+}	p_{2+}	p_m	p_{max}	c_{1+}	c_{2+}	c_m	c_{max}	p_1c_1
g4	161777	175536	1	141615	287	54	175610	740	43	1	196	161105	1071	1	226	68
g2	109634	1390120	2	19186	89168	69310	1597194	71258	24631	1	32376	78874	46779	2	18577	40498
g5	45876	54090	1	34853	2837	418	54658	44349	682	1	1359	2763	1320	1	1481	1236
g15	45706	71451	2	25782	1246	974	89295	13765	132	1	954	44313	3556	1	941	12372
g6	24032	91981	2	15009	4211	680	102153	15944	7330	1	281	19272	10234	6	3325	11184
g10	12614	27443	2	5958	5348	4259	27840	2019	988	1	4104	11047	1401	1	4129	452
g9	11683	23610	1	5736	1206	870	27180	5242	929	1	63	7585	2748	1	311	1144
g3	11214	42501	2	7645	11741	9611	48788	9402	2917	1	1968	8139	4603	2	8632	6327
g13	4517	16163	2	3025	1283	307	20206	2533	2055	3	47	4181	3280	8	337	2197
g8	3147	12382	5	2233	324	134	12440	406	99	1	119	2886	2221	9	64	145
g12	2145	3912	1	488	15331	10856	4369	1357	493	1	7615	1242	459	1	11424	454
g1	1263	97997	18	555	14857	13966	109554	908	601	6	697	779	628	4	6366	424
g17	525	830	1	247	107	47	871	128	37	1	62	446	144	1	55	49
g14	513	773	1	57	13355	201	822	168	50	1	12422	419	113	1	12422	74
g20	286	924	2	125	88	60	979	137	45	1	17	203	166	3	25	54
g18	270	465	2	216	34	19	467	43	15	1	9	232	196	3	20	5
g7	261	583	2	89	102	31	611	219	20	1	18	57	22	1	62	15
g16	214	275	1	107	61	44	283	102	27	1	26	124	69	2	23	12
g11	199	1522	7	98	61	38	1549	148	34	1	12	91	68	3	22	40
g19	79	133	1	44	32	24	133	66	16	1	10	18	7	1	21	5
g21	52	1282	50	51	2371	1398	2507	52	51	8	694	50	50	13	1161	50
g21'	52	682	21	51	2369	1396	952	52	51	6	692	50	50	11	1159	50

Table 1: Size statistics on the 21 graphs, shown in order of number of nodes. The columns are: the number of nodes $|V|$, the number of undirected edges $|E_u|$ (ignoring different ports as well as edge direction), the median undirected degree d_m, the maximum undirected degree d_{max}, the number of unique (service) ports over the entire graph (note: each directed edge is associated with one or more service ports), the number of such ports used by more than one node pair (ports2), the number of directed edges $|E_d|$ (ignoring different ports), the number of nodes providing one or more service ports p_{1+}, the number of nodes providing two or more service ports p_{2+}, the median p_m (over those providing at least one port) and max p_{max}, and the same for client (consumer-of) counts (c_{1+}, c_{2+}, c_m, c_{max}), and finally the number of nodes providing at least one port and consuming at least one port, p_1c_1 (equivalently, both indegree and outdegree ≥ 1). Graph g4 is the largest in terms of number of nodes (at 162k nodes), but other statistics (*e.g.* ports, p_{1+}, p_1c_1) reveal that it effectively has smaller rich structure than the raw number of nodes might imply. g21 is the graph with a ground truth grouping. For this graph, port 1 under both TCP and UDP protocols is highly frequent, and we also present the statistics when those two ports are removed, in row g21'.

connection, and degree, and thus there may be some inaccuracies. Turcotte et al. also describe three heuristics they use for orienting the flows and deduping in the host data they provide (they keep the client port) [16]. We estimate the accuracy of server designation to be in the high 90s (as percentage of edges) based on our various validation experiments. The other (client) port is often dynamically chosen and changes frequently (ephemeral), and is uninformative. We do not report the client port (including the client port would also substantially increase the number of unique edges to report as well). We note that there are applications such as Hadoop, where server ports can also change frequently (see Section 2.4),[4] and other applications (such as NTP) where client port can be fixed too.

[4]As we don't in general know the applications running on a customer's network, and how they are configured, our pipeline can detect *port-intervals* (from which dynamic service ports are picked) and estimate the extent of the port intervals. These problems are variants of serial estimation (population estimation) or the so-called German Tank problem [13]. Robust estimation of port intervals is important for micro-segmentation and similarity computations or feature encoding (Sec. 2.4).

2.1 Size Statistics

Table 1 reports on a range of graph statistics. Both the number of directed and undirected edges (*i.e.* when port and edge direction is ignored) are reported, and we observe that most edges (interactions) are in one direction, as the number of directed edges is not substantially higher than the undirected count, indicating asymmetry of interaction. Table 2 includes the number of directed and port-differentiated edges. For a combination of reasons (such as external/periphery nodes, the extent of sensor deployment and time of sensing, some nodes being stand-by), many nodes have one or very few edges. We report the "2+" versions of several statistics (number of nodes providing more than 2 ports p_{2+}, and number of service ports associated with at least two edges, ports2), to better convey the activity level. The peel1 column in Table 3 indicates the number nodes with (undirected) degree 1. We also observed that one or a few nodes have very high degree d_{max} in some graphs, suggesting the possibility that they provide a (graph-wide) common service (Section 2.2.1).

G	$\lvert E_{d,p}\rvert$	once	twice	all batches	ports & nodes
g4	175985	82%	6%	2255, 1%	41, 169, 1521
g2	6964428	87%	4%	76198, 1%	1590, 7491, 16721
g5	61828	46%	33%	1280, 2%	42, 402, 331
g15	95774	44%	16%	298, 0%	12, 98, 190
g6	308505	39%	7%	39761, 12%	66, 8317, 9584
g10	112971	50%	16%	3550, 3%	115, 540, 775
g9	53489	47%	8%	1619, 3%	71, 408, 352
g3	162005	49%	12%	10417, 6%	96, 898, 1988
g13	61448	26%	11%	178, 0%	9, 12, 91
g8	33066	15%	4%	4352, 13%	35, 63, 1613
g12	39772	78%	10%	1353, 3%	29, 292, 323
g1	1130224	51%	15%	187328, 16%	306, 545, 462
g17	1203	17%	4%	55, 4%	15, 13, 11
g14	14320	90%	3%	339, 2%	37, 76, 170
g20	1373	14%	3%	4, 0%	4, 3, 1
g18	724	10%	0% (5)	335, 46%	13, 16, 190
g7	767	19%	6%	422, 55%	19, 154, 47
g16	415	17%	8%	165, 39%	36, 46, 72
g11	1736	8%	3%	873, 50%	26, 90, 37
g19	258	30%	7%	46, 17%	12, 19, 12
g21	9966	18%	6%	1917, 19%	89, 51, 50

Table 2: (Persistence of Edges) The number of port-differentiated directed edges $\lvert E_{d,p}\rvert$ (compare to $\lvert E_d\rvert$ in Table 1), and percentages of such edges as a function of edge persistence (frequency): percentage of those observed once (in one file or recorded batch), twice, and in all files (13 files for all graphs except g21, 96 files for g21). For "persistent" edges (occurring in all files), both the counts and percentages are shown. The last column reports respectively the number of (unique) service ports, server nodes, and client nodes responsible for the persistent edges. Thus 41 ports, 169 different provider nodes, and 1.5k client nodes accounted for the over 2k persistent edges in g4, and in g4 about 1% of edges are persistent. We observe that most edges in most graphs occur once or few times, but the fraction of persistent edges can be significant too (possibly yielding a concave histogram). Often, a relatively small number of ports and nodes account for all the persistent edges. The graphs are shown in the same order of Table 1.

G	max ind	in-port	max outd	out-port	peel1	peel2
g4	87%	1p6	0%	4p6	92%	99%
g2	17%	5p17	2%	2p6	45%	45%
g5	1%	7p6	75%	2p17	92%	97%
g15	56%	11p17	20%	12p17	47%	95%
g6	61%	1p17	46%	3p17	33%	97%
g10	46%	2p17	2%	6p17	48%	89%
g9	46%	1p6	27%	1p17	75%	96%
g3	35%	10p17	67%	1p17	34%	63%
g13	48%	5p17	23%	5p6	22%	51%
g8	66%	6p17	1%	4p6	23%	85%
g12	21%	22p17	20%	3p6	62%	62%
g1	23%	3p17	20%	210p6	25%	25%
g17	46%	14p6	4%	3p6	58%	58%
g14	9%	4p6	1%	5p6	65%	65%
g20	43%	6p17	7%	1p17	23%	23%
g18	80%	1p17	4%	4p17	24%	100%
g7	7%	5p17	34%	1p6	24%	24%
g16	19%	1p6	12%	17p6	76%	100%
g11	16%	7p6	14%	16p6	2%	2%
g19	6%	24p6	20%	10p6	58%	100%
g21	96%	6p6	96%	18p6	1%	100%

Table 3: (Max Degree and Peelability) The maximum in-degree (max ind) with the same port, and outdegree (max outd), shown as a percentage of the number of nodes in the graph. Thus for g4, a single node provides the port 1p6 (port number 1 (de-identified), TCP protocol), to 87% of the nodes in the graph. The fractions of nodes removed under two peeling regimes are also shown (edges treated as undirected here). Under peel1, degree 1 nodes are removed (repeating did not yield further pruning). Under the more aggressive peel2, degree-1 nodes as well as nodes connecting to at least 60% of remaining nodes are removed repeatedly.

The number of unique ports and ports2 (*i.e.* 2 or more node pairs using same port) are a reflection of the extent of different types of tasks in the networks. However, as noted above, there are some applications (*e.g.* in g21) in which service ports also change frequently. We next describe some brief explorations of processing the graphs.

2.1.1 Edge Frequency. The edges are organized into hourly dumps and thus, to an extent, one could also explore graph evolution or take edge frequency into account in a task such as graph partitioning. Each file contains edges occurring for some duration inside one hour, and the same edge may appear in multiple files, signaling frequency of usage. Table 2 presents the number of directed port-differentiated edges,[5] and percentages of those counts seen in one, two, or all (13) files. Often, most edges are seen once (infrequent or short-lived), but observe that the percentage of such varies widely among different graphs. The last column reports the number of (unique) ports, service nodes, and client nodes, in that order, behind the (most) persistent edges. We observe that specially the number of ports relative to the number of persistent edges is small, and this is also the case, but to a lesser degree, for the number of provider and client nodes. These patterns are not uniform across datasets.

The dataset also contains the number of packets exchanged, but information such as the duration of the connection is not reported (nor payload contents).

2.2 Preprocessing

We provide the graphs as described above, *i.e.* without further preprocessing. However, preprocessing can be essential for a more

[5]In a graph of two nodes u and v, if u is client of v on 2 ports, and v is a client of u on 1 port (a directed 2-cycle), then $\lvert E_{d,p}\rvert = 3$ (Table 2), $\lvert E_d\rvert = 2$ and $\lvert E_u\rvert = 1$ (Table 1).

effective downstream analysis such as community detection and clustering. Furthermore, preprocessing can uncover common services, such as a service port used by many nodes (node pairs), as well as nodes that connect to a large portion of the network and perform a common task (nodes with large out-degree often providing the same service port). This high in-degree or out-degree may reflect a disassortative property of several graphs or parts of such graphs, and the opposite of the common (assortative) community property (see Section 2.3). Finally, basic preprocessing algorithms can help provide insights into the structure of the observed graph, *e.g.* what the external nodes might be, or where there is limited visibility due to lack of sensors. Note that for an edge to be reported, a (software) sensor must be placed on (at minimum) one of the two ends of the edge, and customers may deploy sensors for only a subset of their network (an exception is when hardware sensors are also present). Furthermore, sensors can be tuned off, or may not report all the traffic in certain special cases, *e.g.* due to requirements that sensors should not consume too much computational resources.

2.2.1 Common Services. Table 3 shows, as a percentage of number of nodes, the maximum indegree and outdegree, where both port and direction are considered, thus if node u provides port 1 to two nodes, and provides port 10 to three other nodes, its (maximum) indegree is 3 (and not 5), and we take the maximum of the indegrees over all nodes. The protocol and the port number yielding the maximum is also shown in the table. Note that the port number reported is not the actual port number, but an integer assigned (for example, port TCP 80 may be mapped to TCP port 1). This mapping is performed separately for each graph and each of the two protocols.[6] In the table, a port number say 11 under protocol 17 (UDP) is shown as 11p17.

We observe that a number of datasets have nodes with very high fan-in (common service). There can be multiple nodes with high degrees, and we are reporting the degree of the highest. Interestingly, there also exist nodes with very high out-degree for several graphs (consumer of the same fixed port, but from different provider nodes). While port 1 is the most common port for g21, port 6p6 yields the highest indegree.

2.2.2 Peeling. We next report on what we refer to as graph *peeling*, *i.e.* repeatedly removing nodes that have either high degree (such as connecting to more than say 60% of nodes), or very low degree, *e.g.* having only 1 edge. Once such nodes are removed after a first pass, some of the remaining nodes may now also satisfy the criteria, *e.g.* a degree 3 could become a degree 1 node after removing two of its neighbors (or may become 'orphaned', degree 0, if all its connections are removed), and we can repeat this "peeling" (removal) process, until all the remaining nodes have sufficiently high connectivity and at the same time not have too many edges (*i.e.* unlikely to provide a common service).[7] We find that workload g4 (from Table 1) is highly peelable, as removing those with degree 1 yields a graph of just 12k nodes. If we then remove node(s) that have high degree, connecting to $\geq 60\%$ of nodes remaining, and repeat the process of removing those nodes that satisfy either criterion (degree 1 or above 60% connectivity), the final remaining graph has only just over 200 nodes with just over 900 edges. On the other hand, g2 is not as peelable: the same process leaves almost 60k nodes and over 1.3 million edges in the final remaining graph. Table 3 shows the fraction of nodes that is removed under two peeling regimes. In both cases, ports and direction are ignored (undirected). In the first regime, peel1, only nodes of degree 1 are removed. We observed no difference if we repeated this process (the graphs did not reduce further). In the second regime, peel2, as described above, high degree nodes (set at 60%) are also removed, and the process is repeated. We observe that several graphs substantially peel.

2.3 Community Detection

Discovered communities in the network can be good candidates for grouping the nodes: the discovered groups can reflect departments, applications and other partitions of the network, such as testing *vs.* production machines. The user can browse and modify the candidates groupings and may rerun the algorithms with different parameters or on different time periods. As mentioned in the introduction, one can use this processed traffic, that is the edges of the graph as well as the groupings, to induce *allow-policies* (security policies), for the end-user to examine and modify, and approve and enforce such policies (assuming enforcement agents are deployed on the hosts). Enforcement of an allow-policy means any other traffic (one that does not match the policy), is disallowed (dropped).

Most community discovery algorithms operate on the intuition that a network community should be dense inside and sparse outside, the assumed assortative property (*i.e.* relatively many edges connecting group members to one another, and relatively few edges connecting the members to external nodes) [5, 14]. We applied the Louvain algorithm [2], on the undirected and unweighted versions of the 20 graphs, *i.e.* two nodes get an edge whenever one is the client of the other, irrespective of port(s) or protocol(s). Louvain is a fast algorithm that uses the Modularity objective [11], and it takes minutes to finish[8] on most of the graphs on a laptop (2 hours on g1). Over all the 20 graphs, a total of almost 300 groups are discovered, with just over 200 with size ≥ 3 (3 nodes or higher), and just over 150 with size ≥ 10. As expected, larger graphs tend to yield more groups, *e.g.* w2 yields over 110 groups. The largest group had over 100k nodes. We can assess the statistical significance (via binomial tails) of the groups discovered, in terms of the number of internal and external edges observed (with respect to a group) and random models of connections as null (reference) models [9], and nearly all have high significance (over 99%) and the number of significant goes down to just over 80 if we adjust for graph size.

Preprocessing, and taking direction and the port information may improve the groups discovered. In particular, we have found that weighting the edges as a function of the uniqueness of the ports used in the communication (*i.e.* informativeness, based on the frequency of usage in the entire graph, akin to tf-idf in information retrieval [1, 7]) can improve the modularity and (binomial) significance scores and, when we have ground truth (next section), we

[6] In this mapping, the most frequent port, because it is likely seen early in the data, is likely assigned a small integer, possibly 1.

[7] It is also possible that a host provides a common service to only a subset of nodes, which can require a hierarchical grouping for discovery. See peel2 in Table 3.

[8] We have also experimented with implementations of the spectral algorithm, one from sklearn.cluster [12], but the algorithm did not converge in several scenarios (or the groups generated were inferior).

also observe improvements in the overlap of the discovered groups with the ground truth groupings. In general, edges have a number of attributes as explained above (direction, number of packets exchanged, ports, persistence), and taking such information into account may improve results or change the outcome in interesting ways.

2.4 A Graph with a Ground Truth

Ground truth groupings can help validate or provide insights into performance of various techniques such as vector representation (feature encoding), similarity computation, and clustering or community discovery algorithms. We provide ground truth groupings for a relatively small graph, g21 in Table 1, which is an instantiation of our own workload. The workload includes flow data processing pipelines (a Hadoop application), various graph algorithms (such as community detection and clustering) as well as other tasks such as Elasticsearch (for flow search, UI). Thus the sensors and the system collect, process, and report their own activity. Our grouping is based on node function (functions such as datanode, nodemanager, orchestrator, mongodb, ...). The ground truth has 21 groups, the largest group has 7 members, and four groups are singletons (one member). Running Louvain, on the undirected unweighted graph, yields two groups with a very low modularity score of 0.002, while if we remove port 1 (1p6 and 1p17), the modularity score improves to 0.06 with 3 groups generated. If we construct a weighted graph based on number of ports used between any pair of nodes and the uniqueness of the ports, modularity further improves. The average Jaccard overlap of the discovered groups with the (best matching) ground truth groups also improves, from 0.06 (no preprocessing) to 0.17 (with weighted edges).

2.4.1 Similarity Computation. Community detection algorithms, when invoked on the original graph of interactions, are not in general appropriate for finding groups of homogeneous nodes (that perform the same function): they tend to find groups of well connected nodes which nevertheless can contain different types of nodes (performing different subtasks). Community discovery may be more appropriate for partitioning at a relatively coarse level, likely more useful for discovering nodes at the top of an organization's tree. Clustering based on an appropriate measure of similarity could be more fruitful for finding relatively small but homogeneous hosts. There are a number of general challenges when discovering groups based on the sensed data. Sensors are imperfect and may not be deployed on all nodes, and not all the behavior of tasks may be reflected in the observed network traffic, thus the data gathered is often noisy and incomplete. Hosts can show diverse behavior, for instance, some may be on stand-by or serve as back up. Finally clustering, and in general unsupervised learning, can be subjective or poorly defined. For instance: should a host that performs both tasks A and (at times) task B be clustered with five other hosts that only perform task A, or does it belong to its own singleton cluster?

We can define similarity between two nodes based on the ports they provide (as server) or consume (as client) as well as possibly taking into account the destination nodes that they are clients of or servers to (or groups, if available). This similarity can then be used to cluster. A simple vector construction for each node based on the above feature encoding ideas and ranking other nodes based on the cosine similarity of their vectors yields an average precision at rank 1 of just over 90%, where the average is over nodes that belong to a ground truth groups of size ≥ 2 (the ground truth groupings are used in computing precision). The average precision performance drops to below 70% when for each queried node u, we compute the proportion of top $|gt(u)| - 1$ ranked nodes (by similarity) that belong to the ground truth group $gt(u)$ (which contains u). Thus, there is room for improvement in terms of similarity computation.

3 CONCLUSIONS

We are providing 21 graphs reflecting host communications underlying diverse distributed applications exhibiting rich connectivity and behavior patterns. We briefly explored a few use cases. We expect the dataset will be useful in exploring graph mining techniques such as clustering and community detection, in particular in the domain of distributed workloads, with applications to network management and security.

ACKNOWLEDGMENTS

We thank Sumit Babu and Ravi Sankuratri for assisting us in getting the data. Thanks to the anonymous reviewers, and many thanks to Roland Yap, who helped us improve the content and presentation.

REFERENCES

[1] R. Baeza-Yates and B. A. Ribeiro-Neto. 2011. Modern Information Retrieval - the concepts and technology behind search, Second edition.
[2] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. 2008. Fast unfolding of communities in large networks. *J. of Stat. Mechanics: Theory and Experiment* 2008, 10 (2008).
[3] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen. 2020. A Survey of Intrusion Detection Systems Leveraging Host Data. *ACM Computing Surveys (CSUR)* (2020).
[4] D. Dua and C. Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[5] S. Fortunato and D. Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (2016).
[6] V. Jeyakumar, O. Madani, A. ParandehGheibi, and N. Yadav. 2016. Data Driven Data Center Network Security. In *Proc. IWSPA*, R. M. Verma and M. Rusinowitch (Eds.). ACM.
[7] K. S. Jones. 1972. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation* (1972).
[8] J. Leskovec and A. Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
[9] O. Madani, T. Ngo, W. Zeng, S. A. Averineni, S. Evuru, V. Malhotra, S. Gandham, and N. Yadav. 2020. Binomial Tails for Community Analysis. arXiv:2012.09968 [cs.SI]
[10] M. Mujib and R. F. Sari. 2020. Performance Evaluation of Data Center Network with Network Micro-segmentation. *12th International Conference on Information Technology and Electrical Engineering (ICITEE)* (2020).
[11] M. EJ. Newman. 2006. Modularity and community structure in networks. *PNAS* 103, 23 (2006), 8577–8582.
[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[13] R. Ruggles and H. Brodie. 1947. An Empirical Approach to Economic Intelligence in World War II. *J. of the American Statistical Association* (1947).
[14] M. T. Schaub, Jean-Charles Delvenne, M. Rosvall, and R. Lambiotte. 2017. The many facets of community detection in complex networks. *Applied Network Science* (2017).
[15] N. I. Sheikh, M. D. Pawar, and V. Lawrence. 2021. Zero trust using Network Micro Segmentation. *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2021).
[16] M. J. M. Turcotte, A. D. Kent, and C. L. Hash. 2019. Unified host and network data set. In *Data Science for Cyber-Security*. World Scientific. https://csr.lanl.gov/data/2017/.