Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier TBD

Analysis of Malware Communities Using Multi-Modal Features

IAIN J. CRUICKSHANK[1], KATHLEEN M. CARLEY[2], (Fellow, IEEE)
[1]Center for Computational Analysis of Social and Organizational Systems, Carnegie Mellon University, USA (e-mail: icruicks@andrew.cmu.edu)
[2]Director, Center for Computational Analysis of Social and Organizational Systems, Carnegie Mellon University, USA (e-mail: kathleen.carley@cs.cmu.edu)

Corresponding author: Iain J. Cruickshank (e-mail: icruicks@andrew.cmu.edu).

This work is supported in part by the Office of Naval Research under the Multidisciplinary University Research Initiatives (MURI) Program award number N000141712675, Near Real Time Assessment of Emergent Complex Systems of Confederates, the Minerva program under grant number N000141512797, Dynamic Statistical Network Informatics, a National Science Foundation Graduate Research Fellowship (DGE 1745016), and by the center for Computational Analysis of Social and Organizational Systems (CASOS). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR or the U.S. government.

ABSTRACT Identifying possible threat actors from samples of malware remains an active area of research with important ramifications for cybersecruity practitioners. The unsupervised identification and characterization of malware samples has been primarily treated as an early integration, multi-modal clustering problem where all possible features derived from the samples are concatenated into one feature vector, which can then be fed into a standard unsupervised learning algorithm. In this work, we focus on characterizing malware samples into possible threat actors from both late integration and intermediate integration multi-modal data perspectives. In doing so, we propose a new algorithm, Cross-Modal Influence Clustering, for characterizing malware samples into threat actor groups. We test our proposed method along with several other multi-modal clustering techniques on heterogeneous malware samples from three different threat actors. Our results indicate that our proposed method is the best method for clustering malware samples into threat actors in an unsupervised setting and that we observe consistently better results in characterizing malware samples by threat actors from intermediate or late integration multi-modal paradigms rather than an early integration one.

INDEX TERMS Cybersecurity, Malware, Multi-modal data, Network Diffusion, Unsupervised machine learning, Multi-view data

I. INTRODUCTION

MALWARE continues to be one of the most prolific and destructive threats to cyber systems. Malware is also often a tool of choice for threat actor groups that seek to compromise various cyber systems [1]. The volume of malware produced continues to accelerate resulting in thousands of new samples of malware being discovered everyday. The vast majority of malware samples are actually variants of existing malware, produced by transforming or obfuscating an existing sample in such a way that it can evade detection by security products and other defenses. As such, many malware samples could be characterized as being a malware community or family with distinct relations between samples. Furthermore, many malicious actors employ different types of malware (i.e. Worms, Trojans, Viruses, etc.) that have certain distinct patterns in their features that can aid in identifying which actor a malware sample may have come from [2]. So, while there are millions of malware samples, they tend to congregate into families which may only be used by a certain threat actor. Consequently, the categorizing of related malware into possible threat actor groups and understanding relationships between the malware used by different threat actors can aid malware analysts and security operations in prioritizing responses and defenses for new malware-based attacks.

While there are a number of existing techniques for analyzing malware [3], one approach that has not been explored significantly involves analyzing features from malware samples from a *multi-modal*, or *multi-view* perspective. That is to say there are many types of features, modes or views, from static to dynamic or hybrid, that could be used to identify different types malware and possibly characterize them into threat actor groups. Most work on using these various features of the malware to date and the best of the authors' knowledge

take an *early integration* approach with these features. In early integration, all of the possible features derived from the malware are concatenated into one feature vector and then fed to a clustering algorithm. Furthermore, most research on identifying and characterizing malware samples focuses around placing malware samples into malware families; few works to date try to identify the threat actor(s) that are using and developing the malware [3]. In this work, we propose approaching the task of characterizing samples of malware into possible threat actor groups, without any prior knowledge of possible threat actor groups, as an intermediate or late integration multi-modal, clustering problem. Furthermore, we propose a new algorithm, Cross-Modal Influence Clustering, which shows the best performance in our experiments of characterizing heterogeneous malware samples into three, distinct threat actor groups. Our algorithm combines two major lines of research on multi-modal clustering: graph-based, intermediate integration methods and cluster ensembling. In our proposed algorithm, we blend the two methods by using individual modal clusters, as in cluster ensembling, but also use modal graphs to diffuse those cluster labels across modes, which results in better clustering results relative to the threat actors that the malware originated from. We demonstrate both the effectiveness of intermediate and late integration multi-modal approaches to clustering samples of malware into threat actor groups and our proposed algorithm using several hundred heterogeneous malware samples from three different threat actors: Dukes, APT-1, and Deep Panda. Our main contributions in this work are summarized as follows:

- A new algorithm that characterizes malware samples into clusters that gives some indication of the threat actor groups behind the malware.
- Clustering malware samples using intermediate or late integration multi-modal approaches, rather than early integration, consistently shows better results in malware threat actor characterization.
- The performance of the different multi-modal clustering algorithms is demonstrated using heterogeneous malware samples from three different malware threat actors.

The article is organized as follows: In the next section we review some of the work related to characterizing malware families and multi-modal clustering. In Section 3, we outline the two main methods used in this work. The first method is used to obtain graphs and clusters for every mode and the second method is our contributed method of clustering the multi-modal data. In Section 4 we present the results of testing the algorithms on malware data. Finally, the last section concludes with some reflections on our results and presents avenues for future research.

II. RELATED WORK

The characterization and identifications of malware samples remains an active area of research. Many of the current methods today break down into whether they employ supervised and unsupervised machine learning and what types of features they use [3]. In terms of the features being used to characterize or identify malware, most features come from either static or dynamic analyses of the malware [3]. Dynamic features tend to focus on malware behavioral patterns, whereas static features tend to be aspects of the malware sample itself [3]. One of the most promising set of static features from a malware classification standpoint were those proposed by Berlin and Saxe in [4]. These features have been shown to give state of art performance in supervised learning of malware samples [4] as well as being useful for characterizing malware samples within a threat actor group in an unsupervised learning paradigm [5]. Despite the prevalence and variety of malware features available, most approaches in use today use an early integration approach with the different modes of the data. That is to say, current methods take all of the modes of features for identifying the malware and concatenate (typically with some kind of normalization or feature engineering across modes) all modes of the features into one feature vector and then feed those vectors into standard machine learning algorithms.

Multi-modal, or multi-view, data is data which consists of different types of data but all of which are describing the same underlying phenomenon. Many phenomenon of interest naturally give rise to multi-modal data [6]. A common example is film, which can give rise to audio and visual data. Generally speaking both supervised and unsupervised machine learning can benefit from the use of multi-modal data [6]. In this respect, multi-modal data is different from knowledge graphs, which contain variously linked data, but not necessarily data that is all describing the same objects. This is particularly true when the different modes of the data contain complementary information, such that different modes of the data can capture different aspects of the underlying phenomenon. Despite its potential utility for machine learning tasks, it is often not clear how to combine multi-modal data in order to best utilize its benefits [6], [7]. As a result, most multi-modal data is treated in an early integration approach whereby each of the modes of the data are concatenated into one mode and then the problem becomes a traditional uni-modal problem where traditional uni-modal machine learning techniques can apply. Despite their simplicity, early integration approaches often perform less well with multi-modal data then do intermediate or late integration approaches [7], [6]. Late integration approaches use algorithms on each mode and then try to combine the outcomes of each algorithm's results [6]. A common example of these approaches are cluster ensembling techniques [8]. Intermediate integration approaches, in contrast to the other two approaches, use an algorithm that works on all modes simultaneously without their concatenation but often with some transformation of each mode of the data into a particular format, like a graph, so that the data is all in the same space [7]. These techniques often create a joint representation of all of the modes of the data, like a tensor, which can then be used with clustering techniques [6], [7]. All together, the use of multi-modal data in machine learning applications remains an active area research, often with specific algorithms being

developed for specific multi-modal data scenarios.

Within the realm of using multi-modal data for unsupervised learning, there are generally two main areas of methods outside of early integration approaches. In one major area of methods are cluster ensembling methods [9], [8], [10]. These methods typically work by getting the cluster assignments from each mode independently, and then integrate those cluster assignments into a final, more robust clustering for the objects in question. Cluster ensembling methods were traditionally developed for uni-modal data whereby a user would take multiple clusterings of the same data, or subsets of the same data, and not multi-modal data [9], [11]. In that usage, cluster ensembling has seen a wide range of use across data scenarios, to include family characterization in malware [12], [13]. Only more recently has cluster ensembling been applied to multi-modal data, where now each ensemble member is taken from a mode of the data (i.e. a clustering of the image data and a separate clustering of the text data for something like multi-modal image and text data) [8], [14]. In this usage, a recent paper used modified versions cluster ensembling algorithms originally proposed by Strehl and Ghosh [11] to combine some static and dynamic malware features for better malware clustering [13]. Another technique used features from Phishing websites along with the malware present within the Phishing website as to the modes of the data to characterize Phishing websites into communities and even assign attribution [9]. Despite these works, cluster ensembling has not seen as wide of adoption or usage on multi-modal data as it does not take into account complementarity of information between the modes of data when doing clustering, as each mode is clustered independently [15], [16].

The other main class of multi-modal clustering methods outside of early integration approaches are intermediate-approach, graph-based methods. These methods are primarily used for clustering image data, cellular or genetic data, or multi-layer social networks [17], [7], [18]. In these methods, a graph is either constructed or given for each mode of the data. The graph preserves the local similarity of each object being clustered to every other object for that mode and provides a common data format that makes design of a clustering function for optimizing possible. Then, the graphs are collectively clustered. The methods for collectively clustering the modal graphs are often either diffusion processes (see [19], [17], [20] for examples) or tensor decompositions (see [15], [21], [22], [23], [24] for examples). Outside of these two main graph-based methods, there are also graph-based methods that use weighted metrics to combine graphs for clustering [25], [26] as well as weight-learning techniques in spectral paradigms to cluster the modal graphs [27] or combining graph decompositions from across modes to then cluster uni-modally [16], [28]. While the graph-based methods are often considered the predominant means of multi-modal clustering its not clear how well they can do on more general types of data outside of images or social networks, where the graphs of each mode have similar properties to each other. Additionally, the construction of the graphs for each mode is not always an easily solved problem and an active area of research itself [29]. Finally, graph based methods do not take advantage of cluster labels produced within each mode; some types of data, like text, have optimized clustering routines that could produce very meaningful cluster labels that cannot be used in current graph-based methods. So, while intermediate integration approaches have seen a lot of success in dealing with multi-modal data, they can often suffer in performance when the different modes are very heterogeneous making mapping the modes to a shared space difficult [14], [10].

In summation, the characterizations of malware samples whether into families or into more complex groups like threat actors remains an area of active research, and one that uses many different types of data to characterize and identify malware. Additionally, the use of multi-modal data in unsupervised learning also remains an active area of research with many methods either employing a cluster ensembling paradigm or exploiting joint data representations using graphs. There has been little work analyzing malware data from a multi-modal data perspective using other than early integration techniques or exploiting both modal clusters and graphs within the same multi-modal clustering algorithm.

III. METHOD

Our method consists of three, main steps: 1) Extract the different features of the malware using the static analysis from Berlin and Saxe [4]. 2) Find the cluster assignments and nearest-neighbor graphs for each of the modes of the data. 3) Use a cross-diffusion, influence process to update the cluster assignments for each sample, and cluster the resultant bipartite graph for the final cluster assignments. The first steps are existing means of dealing with malware data and graph learning respectively, so we summarize them in the Background section for completeness. The third step of Cross-Modal Influence Clustering is our primary methodological contribution, and is described in subsection B. The following figure, Figure 1, graphically summarizes the entire method for characterizing communities in the malware samples.

Our proposed method uses features of both intermediate and late integration techniques in order to ameliorate the issues with each approach. As in the late integration approach of cluster ensembling, we use the clusters derived from each mode independently, which allows for flexibility to deal with heterogeneity between the different modes of the data. However, in contrast to cluster ensembling techniques, we then use graphs derived from each of the modes to conduct a diffusion process, which is common in intermediate integration techniques. The result of this diffusion process is that the modal clusters, which were initially found independently, are now infused with information from each of the modes. In this way we ameliorate the lack of using complementary information between the modes which is a problem with cluster ensembling and still have a flexible way of dealing with heterogeneous data, which is a problem with many

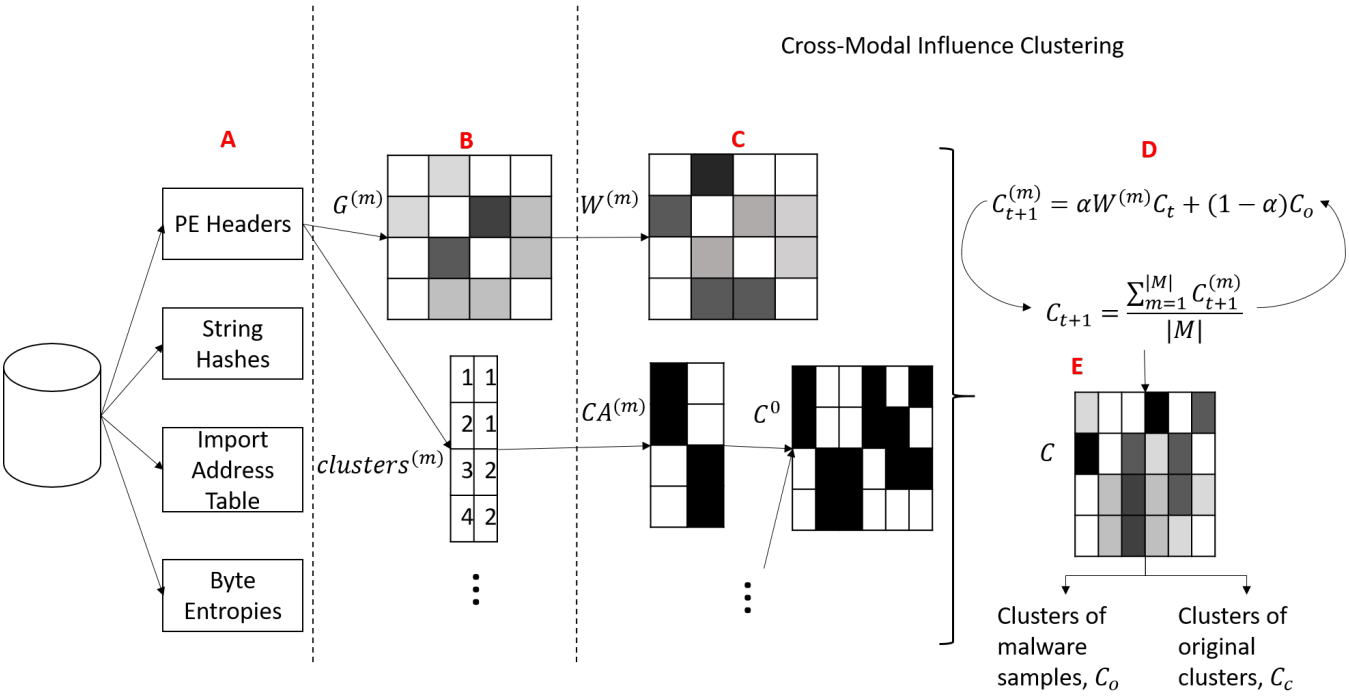


FIGURE 1: Graphical Overview of the Methodology proposed in this work for malware samples into threat actor groups. In step **A** we extract four different sets of features from the malware samples using the methodology originated by Berlin and Saxe [4]. Following the feature extraction, In **B** we use a *Mod-kNN* procedure to find both cluster assignments and nearest neighbor networks, G, for each of the sets of features. Then, in **C-E** we use our proposed methodology, *CMIC*, which is based on concepts from network diffusion and social influence to get the final cluster labels for the malware samples.

intermediate integration techniques.

A. BACKGROUND

Since our approach relies on both clustering assignments and a graph representation for each mode of the data, we detail how modal clusters and graphs are obtained in this section. For the sake of completeness, we also provide information on how the Berlin and Saxe [4] features are extracted from malware samples.

1) Obtaining Features from Malware Samples

The features from Berlin and Saxe come from a deep neural network-based extraction method that uses four static feature domains. In previous works these features have been show to achieve high accuracy malware classification results [4], as well as provide insight into malware communities present in a threat actor group through clustering [5]. The four categories of features are Byte Entropies, Import Address Table, String Hashes, and Portable Executable (P.E.) headers. The Byte Entropies are produced by concatenating the row values of a two-dimensional histogram constructed from byte-entropy value pairs of all bytes in a given 1024-byte sliding window. New entropy values and byte-entropy pairs are computed for each window, which traverses the file using a step size of 256 bytes. The Import Address Table Features are generated by hashing library and library function names to the range $[0, 255)$, and measuring the frequency counts of each resulting hash value. The String Hashes features are produced by by hashing all printable strings of length 6 or more (in the ASCII printable range) to the range $[0, 16)$, pairing each string's hash with the log of its length, constructing a two-dimensional feature vector mapping those hash value / log-length pairs, and then concatenating the rows of the histogram. Finally, the P.E. Headers is produced by extracting numerical features from a file's Portable Executable packaging and aggregating those features into a 256-length array. In total, four different sets of features that are all 256 dimensional vectors of non-negative, integer values are extracted in the first step.

2) Obtaining k-Nearest Neighbor Graphs and Cluster Assignments for Each Mode

Since we are interested in the clusters present in the data, we want graphs (and clusters) that highlight any cluster structure present in the data. One way of doing this is to explicitly learn graphs that emphasize in clusters in the data. So, for the first step of finding clusters and nearest neighbor graphs within each mode of the data, we employ the modularity k-Nearest Neighbor graph (mod-kNN) procedure [30]. At a a high level, mod-KNN method iterates through various possible numbers of neighbors for each vertex, k, and selects that k which produces the most *modular* graph, relative to a null-model,

random graph produced on the same set of vertices. It then outputs this graph and the clustering assignment for each of the vertices that maximizes the modularity. Modularity in this case is the network modularity as described in [31]:

$$Modularity(G) = \frac{1}{2v}\sum_{ij}[A_{ij} - \frac{deg(i) \times deg(j)}{2v}\delta(c_i, c_j)] \tag{1}$$

where v is the number of vertices in the graph, and c are the cluster assignments of the vertices. Since it is known that random graphs can give rise to modular structures, we subtract the modularity obtained from a random graph with the same number of vertices and edges from the modularity value from clustering the derived graph. Since modularity is a measure of how much cluster structure is present in a graph, we are selecting graphs that maximize the cluster structure that may be present in the data. We further incorporate the changes to the original algorithm from other works, namely to use asymmetric k-NN graphs for each value of k (wherein an edge exists between two nodes, u and v, if either $u \in kNN(v)$ or $v \in kNN(u)$) and the Louvain method for clustering the kNNs [25], [26], [32]. Finally, we note that the Louvain algorithm, while very fast for clustering a graph, is also a greedy, stochastic algorithm that can lead to different clustering outcomes for a graph depending on the initialization of the algorithm. So, we also run the Louvain algorithm a number of iterations on each asymmetric kNN graph and evaluate that value of k by the average modularity of each of the iterations of Louvain clustering [32]. This is done to provide for greater stability in both the selection of the kNN graph and the resulting cluster assignments by the procedure. The psuedo-code of our implementation of network construction by mod-kNN is detailed in algorithm 1.

In Algorithm 1 an asymmetric kNN graph is created by doing a point-wise maximization with the transpose of the raw kNN graph which is obtained through any subroutine that finds the k-Nearest Neighbors of each data point given a means of calculating similarity or difference between the data points, $kNN(*,*)$. In this way, the graph becomes undirected which is critical for some algorithms that we investigate in the results section. Additionally, while we have represented the iterations of the different Louvain clusterings of the kNN graph within a for loop, this operation can easily be parallelized for performance. Finally, we evaluate the modularity of each possible value of k by using the average difference in modularity, but select the final cluster assignments for the final k, $k*$, by selecting these assignments from the Louvain iterations that gave rise to the best difference in modularity.

Algorithm 1 Modularity k-Nearest Neighbor Graph

input: Distance or Affinity Metric, s, number of iterations to run Louvain method, l_{iter}
output: Optimal k-Nearest Neighbor Graph, G^*, and sub group assignments $C(G^*)$.
for $i = 1 : \lfloor log_2(n) \rfloor$ **do**
  $k \leftarrow 2^i$
  $G_k \leftarrow kNN(s, k)$
  $G_k \leftarrow maximum(G_k, G_k^T)$
  $G_k^r \leftarrow randomize(G_k)$
  for $l = 1 : l_{iter}$ **do**
    $C_l(G_k) \leftarrow Louvain(G_k)$
    $C_l(G_k^r) \leftarrow Louvain(G_k^r)$
  end for
  $Modularity_k \leftarrow \frac{1}{c_{iter}} * \sum(Modularity(C(G_k)) - \frac{1}{c_{iter}} * \sum(Modularity(C(G_k^r)))$
end for
$k^* \leftarrow \text{argmax}_k\, Modularity_k$
$G^* \leftarrow G_{k^*}$
$C(G^*) \leftarrow \text{argmax}_l\, C_l(G^*)$
return $C(G^*), G^*$

B. CROSS-MODAL INFLUENCE CLUSTERING

Given a graph representation of each mode of the data along with the optimal cluster assignments for each mode, we use a cross-diffusion process [19] in a social influence model [33] to get the final cluster assignments for each sample. Our method, which we call Cross-Modal Influence Clustering (CMIC), iterates through two main steps: 1) update the cluster assignments for each mode by diffusion in a social influence model with the cluster association matrix and modal graphs. 2) Combine the updated cluster assignments from each mode into a new cluster association matrix. Once the iterations are complete, the result is a new cluster association matrix that relates each entity to each of the clusters. This matrix naturally forms a bipartite graph of all of the entities by the cluster labels, which can be clustered by any bipartite graph clustering technique.

For the first step in CMIC procedure, we use a diffusion model inspired by social influence models. Social influence models are often used to simulate the propagation and formation of belief clusters within social groups [34], [33]. As we are interested finding clusters of malware samples, we use modal clusters as 'beliefs' and the modal graphs as the means by which samples may influence each others' beliefs. In particular, we use the Friedkin social influence model described in [33] due to its demonstrated utility and stability in terms of outcomes. Friedkin's social influence model is given by:

$$C_{t+1}^{(m)} = \alpha W^{(m)} C_t + (1 - \alpha) C_0 \tag{2}$$

where C is an object by cluster association matrix of size number of malware samples by total number of clusters across all modes. A cluster association matrix is an object by cluster matrix whose entries represent how strongly a particular object is related to a particular cluster. $W^{(m)}$ is the row-normalized modal graph from mode m and α is a influence parameter that balances the strength of influence

over belief in cluster labels coming from neighbors in a mode versus the initial beliefs in clustering labels. It should be noted that this influence model closely resembles those used in metric learning, especially for image retrieval [35].

After having performed the influence step for each mode, we then need to combine the modal cluster association matrices, $C_{t+1}^{(m)}$, to create a singular cluster association matrix C_{t+1}. To do so, we adopted the simple averaging technique used in cross diffusion [19] and bipartite graph partitioning for ensembling [36]:

$$C_{t+1} = \frac{\sum_{m=1}^{|M|} C_{t+1}^{(m)}}{|M|} \tag{3}$$

where M is the collection of modes and $|M|$ is the number of modes. The cross diffusion process diffuses information (i.e. cluster associations) across a graph in each of the modes separately and then uses that newly diffused information in all of the other modes from the one which produced it for another diffusion step [35]. In this way information from each mode is spread across all of the modes, and the complementarity of information between the modes is better used then trying to use each mode independently. The two steps of influence and aggregation are repeated for a specified number of steps and the resulting cluster association matrix, C_{final} is then clustered as a bipartite graph as has been done in other cluster ensembling techniques [36], [37]. The pseudo-code of the CMIC procedure is detailed below, in Algorithm 2

Algorithm 2 Cross-Modal Influence Clustering

input: Graph for each mode $G^{(m)}$, cluster assignment for each mode $clusters^{(m)}$, Social influence effect, α, and number of iterations, $iterations$
output: Cluster assignments for objects, C_o, and for the original clusters, C_c.
for $m = 1 : |M|$ **do**
  $C^{(m)} \leftarrow cluster\_association(clusters^{(m)})$
  $W^{(m)} \leftarrow row\_normalize(C^{(m)})$
end for
$C^0 \leftarrow concatenate(C^{(m)} \forall m \in M)$
$C \leftarrow C^0$
for $t = 1 : iterations$ **do**
  for $m = 1 : |M|$ **do**
    $C^{(m)} \leftarrow \alpha \times W^{(m)} C + (1 - \alpha) \times C^0$
  end for
  $C \leftarrow \sum_{m=1}^{|M|} \frac{C^{(m)}}{|M|}$
end for
$C_o, C_c \leftarrow bicluster(C)$
return C_o, C_c

Input to the algorithm are the graphs and clusters for the different modes (i.e. the output of the mod-kNN procedure), the number of iterations (we generally find 10 to 30 iterations to be sufficient), and a social influence parameter, α. The social influence parameter in a traditional social influence model regulates how much each agent is influenced by its neighbors (i.e. the value of α) in the network versus being influenced by their own beliefs [33]. Typically this value is set to around 0.9 and we have found that around 0.9 works well for CMIC as well. As a first step in the algorithm we create cluster association matrices C^m for each mode of the data m for all of the total number of modes $|M|$. It is row-stochastic by construction ($\sum_i C_{ij} = 1, \forall j$), and in the case of crisp clustering, it will also be a binary matrix with only one entry per row being equal to 1 and all others being equal to 0. We also transform each graph into a transition matrix by making it row-stochastic ($\sum_i W_{ij} = 1, \forall j$). From there, all of the modal cluster association matrices are concatenated into one cluster association matrix C, which is of size number of entities by total number of clusters across all modes. This cluster association matrix then undergoes an iterative two-step process that results in the final cluster association matrix. In the first step, we use a social influence model to update the cluster associations within each mode of each object by a convex combination of its neighbor's cluster associations and its original cluster associations, $C_i^{(m)} = \alpha \times \sum_{j \in N_m(i)} W_{ij}^{(m)} C_j + (1-\alpha) \times C_i^0$, where i is a particular object and j are the neighbors of that particular object in a particular mode, m. Following this step, we then combine all of the social influence updates into one cluster association matrix. This cluster association matrix then becomes the input to the next iteration of the social influence model. After all iterations are finished, we then use a biclustering algorithm such as Bi-Louvain [38], to cluster the final cluster association matrix. In this way, we follow the same procedure as employed in cluster ensembling, where the object by cluster association matrix is clustered to produce a final, more robust clustering [37], [9]. The outcome is then not only the final cluster assignments for each objects but cluster assignments for the original clusters across all modes.

IV. RESULTS

In this section, we analyze the community detection properties of our proposed algorithm, CMIC, against several other multi-modal clustering techniques. We also analyze the graphs produced from the various modes of static features of the malware samples. Finally, we look at the multi-modal nature of the malware data and the robustness of the CMIC algorithm to changes in the user-set hyper-parameters. In order to empirically evaluate the various algorithms we used heterogeneous malware samples from three different threat actors. The first malware threat actor is Deep Panda, which is a nation-state backed actor who targets a wide range of industries, including government, defense, financial and telecommunications [39]. In our data all of the Deep Panda malware samples belong to the Sakula malware family which are variations of remote access Trojan (RAT) tools. [40]. The second set of samples comes from the APT-1 threat actor which uses various types of malware to attack many different systems in government and industry [2]. The third set of samples comes from the Dukes threat actor which

uses various types of malware to attack various targets in support of espionage and security policy decision making for a nation-state [41]. In total, empirical testing was done with 2,185 different malware samples with 288 from the Deep Panda group (Sakula malware family), 973 coming from the APT-1 group, and 924 coming from the Dukes group. Within the the Dukes samples, 83 of the samples have the actual malware family that the sample belongs to (e.g. CosmicDukes, PinchDukes, etc.). It should be noted that the types of malware present (i.e. Trojan, Worm, Virus, etc.) are not necessarily homogeneous within any given threat actor's samples. APT-1 and Dukes contain a greater variety of types of malware than do the Deep Panda threat actor which just has samples from the Sakula family of malware .

A. COMPARED METHODS

In order to get as thorough analysis as possible of detecting threat actor groups from heterogeneous samples of malware, we have have chosen to use several different multi-modal clustering techniques. The intermediate and late integration multi-modal techniques we investigated are as follows:

- **CMIC**: Cross-Modal Influence Clustering is our contributed method. We ran the algorithm for 30 iterations and used $\alpha = 0.9$.
- **CG**: Consistent Graph [24] aims to learn both consistent and inconsistent graphs across graphs from from multiple views of the data. The method does this using graph representations of each of the modes of the data and a unified optimization framework. The final consistent graph output is clustered via spectral clustering for the final clusters.
- **ETL-MSC**: Essential Tensor Learning for Multi-view Spectral Clustering [23] uses a regularized tensor decomposition method along with some other tensor operations to get a lower-dimensional representation of the data. Each mode of the data is converted into a graph and then each graph into a probability transition matrix, which are then stacked to produce a tensor. The output of the decomposition is then clustered by spectral clustering for the final clusters.
- **BGPA+**: Bipartite Graph Partitioning Algorithm [9], [36] is a cluster ensembling technique that forms a bipartite graph from every object and every cluster across all modal clusterings. This bipartite graph can then be clustered to get final cluster assignments. We improve upon existing algorithms by using a bipartite graph-specific clustering algorithm rather than a generic graph clustering algorithm.
- **NFC-CCE**: Network Fusion for Composite Community Extraction [28] uses a Non-Negative Matrix Factorization for a graph of each mode and across each modes in a combined model to produce a low-dimensional cluster indicator matrix which can be clustered via k-Means. Each mode of the data is converted into a graph for input into this method. We used $\alpha = 0.0001$ which was determined by hand tuning the parameter to best fit the malware data.
- **LWMC**: Locally Weighted Meta-Clustering [37] is a cluster ensembling technique that clusters a specially-weighted cluster-to-cluster graph in order to produce meta-clusters which can then be used to determine the final cluster assignments. The weighting in the cluster-to-cluster graph is done by determining the joint entropy for each cluster with respect to every other cluster, across all modes, and represents how well that particular cluster is at capturing an actual cluster in the data. This method closely resembles what has been employed in late integration approaches with malware family identification [13].
- **LWBG**: Locally Weighted Bipartite Graph clustering [37] is a cluster ensembling technique which clusters a specially weighted bipartite graph of objects-to-clusters from all of the modes. The weighting in the object-to-cluster bipartite graph is done by determining the joint entropy for each cluster with respect to every other cluster, across all modes, and represents how well that particular cluster is at capturing an actual cluster in the data.
- **GP-MGLA**: Graph Partitioning with Multi-Granularity Link Analysis [42] is a cluster ensembling technique that creates a weighted graph of all of the objects and the clusters. The weight in the graph varies based on the type of the vertices of the edge (i.e. whether its a object to cluster or cluster to cluster). This weighted graph of objects and clusters is then clustered as a standard weighted graph. We set $\alpha = 0.5$ and $\beta = 2$ as was done in the method's original paper.
- **SPSL**: Self-Paced Spectral Learning [27] iteratively clusters a weighted combination of the graphs from each of the modes, and updates the weight for each mode's graphs depending on how well that particular modal graph does with the clusters obtained by partitioning the weighted combination of all of the modal graphs. Each mode needs to be converted into a graph representation for the method.
- **MCLA+** Meta-Clustering Algorithm [11] is a cluster ensembling technique that computes a weighted cluster-to-cluster graph in order to produce meta-clusters which can then be used to determine the final cluster assignments. In our implementation, we use Louvain instead of METIS to cluster the cluster-to-cluster graph and weight the cluster-to-cluster graph by the mutual information between clusters rather than the Jaccard Index, as these modifications produced better results.
- **ResK** RESCAL Factorization with K-Means Clustering [15] creates a tensor from all of the graphs from each of the modes and then uses RESCAL tensor factorization of this tensor to produce a cluster indicator matrix. The method then clusters the cluster indicator matrix by k-Means to produce the final clusters.

For all of the multi-modal techniques the cluster assignments and graphs for each mode were determined using the mod-kNN procedure, Algorithm 1. This ensured that each technique was using the same starting basis for a fair comparison and reasonable versions of both the modal graphs and modal cluster assignments. For these experiments, the mod-kNN similarity metric used was the Bray-Curtis dissimilarity [43], given by:

$$s_{ij} = \frac{|X_i - X_j|_1}{|X_i + X_j|_1} \tag{4}$$

where i and j are two different objects (malware samples) and X_i and X_j are their respective feature vectors. We also tested Euclidean distance, cosine distance, and the Jaccard Index, but found Bray-Curtis to give the best results for the mod-kNN procedure with the Berlin and Saxe malware features on these malware samples.

B. GRAPHS LEARNED FROM MALWARE FEATURES

The graphs formed by the mod-kNN procedure from the four different types of malware features have distinctly different topologies. In order to characterize the topologies the following table, Table 1, summarizes the graph measures for each of the modes of the malware data.

Overall, all of the modes have strong clustering coefficients, which correlates with a strong cluster structure being present in the graphs [31]. The graphs are, however, very different in topology with respect to the degree assortiativity, average degree, and density, despite all being formed by the same process. These differences in edges statistics would indicate that there are different patterns of structures present within each of the modes. So, combining all of these features together could obfuscate these modal patterns which could be useful for detecting clusters in the samples. So, this would suggest that the use of multi-modal clustering should be appropriate for these malware samples. To get a better idea of the differences in the graphs between the different modes of the malware, the following figure, Figure 2, depicts the actual modal graphs.

The different modal graphs display distinctly different topologies. However, Across the different topologies, some distinct patterns are easily observable from the images. For one, the Deep Panda malware samples, which are all of the Sakula virus, have two distinct clusters in every mode. This suggests that even though the Deep Panda samples are all one homogeneous family of malware, that this malware has some distinct differences within the family. This result is largely consistent with other analysis of the communities in Sakula virus family of malware [5]. Also, in all modes except the Byte Entropies, the Dukes malware also displays a distinct, large community. Thus, the different malware manifest community structure differently across the different modes.

C. COMMUNITY DETECTION RESULTS

In this section I analyze the ability of the various techniques to cluster the malware samples into threat actors. Since all current research on malware data represents the samples' features as being uni-modal (i.e. takes an early integration approach), we have also chosen to test various common uni-modal techniques after concatenating all of the modes of features into one mode. In particular, we create a uni-modal feature representation of each malware sample by simply concatenating the four Berlin and Saxe feature vectors into one feature vector.

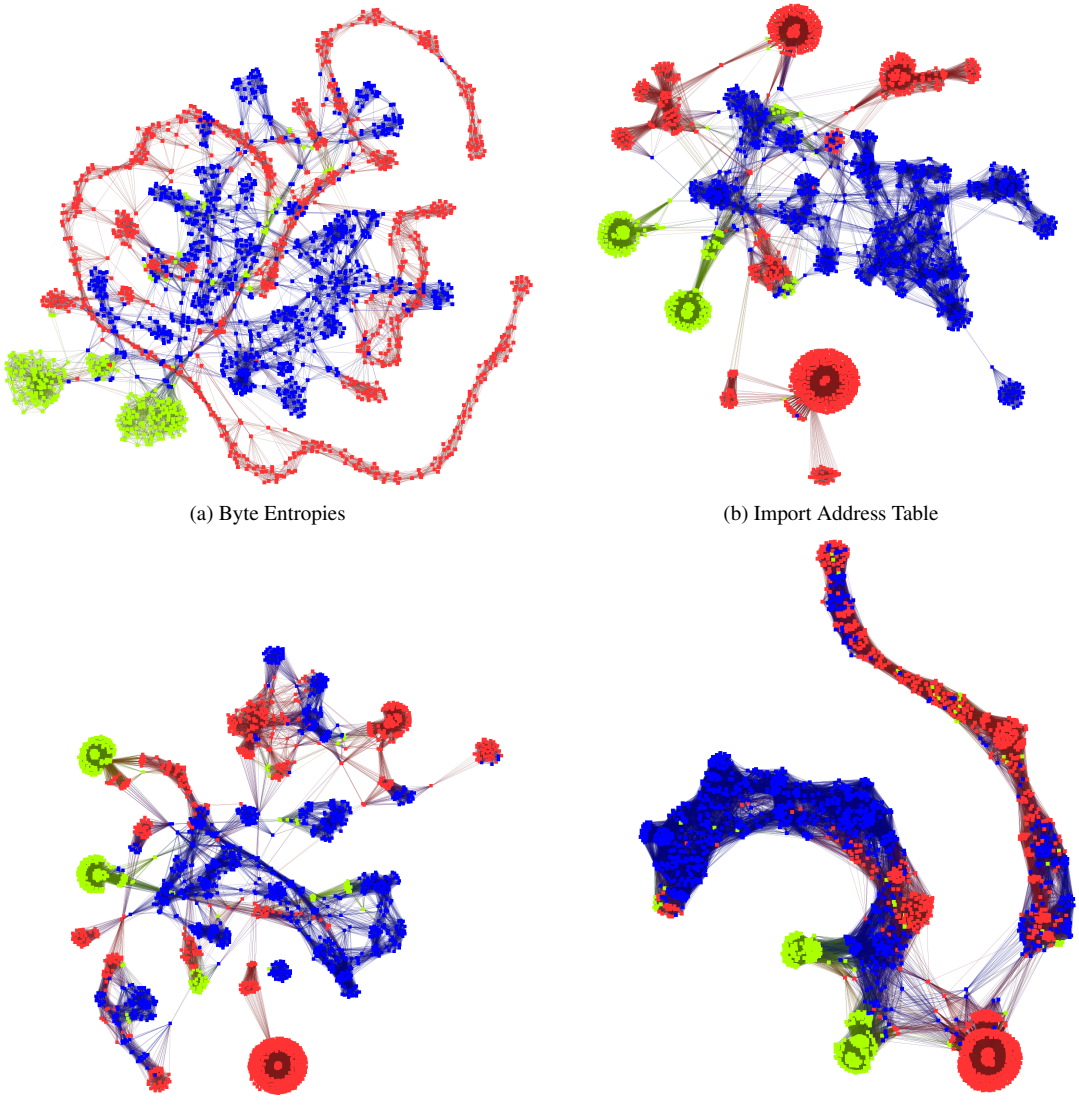
To evaluate the success of the different methods we have chosen to evaluate the clustering outputs by the Adjusted Mutual Information (AMI) [44] and Adjusted Rand Index (ARI) [45] of each method's final clustering with the ground truth threat actor labels of each malware sample. Both of these measures give an indication of how well each clustering matches the ground truth clustering, with higher values indicating better performance. Also, we measured each method's Silhouette score, which is a measure of both how compact and how separate each cluster is, given the original feature vectors [46]. The Silhouette score does not rely on ground truth knowledge of the cluster assignments and is thus regularly used as a means of assessing the goodness of a clustering method in the absence of ground truth labels. Finally, we run each method 20 times and report the average of all of their scores. The following table, Table 2 summarizes the results.

The clustering goodness test presented several interesting results. First, our proposed method, CMIC, performs the best in terms of finding clusters that agree with the malware family labels. Its performance is closely followed by those techniques designed for image clustering and rely on graphs for each mode of the data, namely Consistent Graph (CG) and Essential Tensor Learning for Multi-view Spectral Clustering (ETL-MSC). Each of these are intermediate integration, multi-modal clustering techniques. However, our CMIC method along with some of those in the cluster ensembling, do not require the specification of the number of clusters as does CG, ETL-MSC, and the spectrally-based methods. Thus, our purposed algorithm is able to group the malware samples along threat actor lines heterogeneous types of malware, without knowledge of how many families may be present in all of the malware samples.

Second, in every case, choosing to represent and cluster the malware samples with intermediate or late integration approaches as opposed to early integration gives better clustering results. This is a surprising result as there is nothing *a priori* that would suggest treating malware in a multi-modal fashion would provide for better malware threat group identification. Furthermore, Silhouette does not give a good indication of a clustering method's performance across the multi-modal techniques. When compared to early integration, uni-modal silhouette scores the clusters found by the multi-modal techniques have almost universally worse Silhouette scores indicating that they should have universally

Feature	Density	Number of Components	Avg. Clustering Coefficient	Isolates	Average Degree	Degree Assortiativity
Byte Entropies	0.001	1	0.68	0	21.3	0.14
Import Address Table	0.022	1	0.79	0	47.8	-0.17
String Hashes	0.042	1	0.69	0	92.2	-0.084
PE Headers	0.021	2	0.79	0	46.6	-0.122

TABLE 1: Summary of the topological properties of the learned graphs from each of the views of the malware. In each case, each graph has relatively string community structure, but also wide differences in edge statistics. This would indicate that the patterns are different between the views of features.



(a) Byte Entropies

(b) Import Address Table

(c) PE Headers

(d) String Hashes

FIGURE 2: Graphs of the four different modes of the malware samples. Vertices in red are the Dukes malware samples, blue the APT-1 malware samples, and green the Deep Panda malware samples. In all of the modes, the Deep Panda samples display two distinct communities.

worse cluster assignments. However, our tests clearly show intermediate and late integration, multi-modal clustering algorithms as being better able to find threat actor groups in the samples. Furthermore, there is also no trend in the

Clustering Technique	ARI	AMI	Silhouette
CIC	**0.35**	**0.39**	-0.47
CG*	0.32	**0.39**	-0.49
ETL-MSC*	0.3	0.35	-0.4
BGF+	0.27	0.32	-0.52
NF-CCE*	0.27	0.27	0.004
LWMC	0.24	0.28	-0.7
LWBG	0.23	0.29	-0.71
GP-MGLA	0.23	0.3	-0.55
SPSL*	0.24	0.27	-0.22
MCLA+	0.21	0.26	-0.42
ResK*	0.21	0.25	-0.29
K-Means*	0.18	0.17	**0.97**
Ward Agglomerative*	0.18	0.17	**0.97**
mod-kNN	0.12	0.22	0.05
OPTICS	0.01	0.14	0.177
Spectral*	0.001	0.004	-0.39

TABLE 2: Clustering goodness scores for different clustering methodologies on malware samples coming from three different threat actors. The top partition are late and intermediate multi-modal clustering techniques while the bottom are uni-modal clustering techniques, performed on early integration of the modal features. Techniques denoted with a * require the number of clusters as input

Silhouette scores within the intermediate and late integration, multi-modal techniques; more positive Silhouette scores do not indicate better clustering results for these techniques. So, the cluster goodness testing indicates that our proposed method performs the best at identifying meaningful threat actor labels from malware samples and that intermediate and late integration, multi-modal techniques always outperform early integration with uni-modal techniques in this clustering task.

D. ANALYSIS OF DISCOVERED COMMUNITIES

Having demonstrated the usefulness of CMIC in finding threat actor groups from samples of malware data, we then analyzed the communities found by the CMIC method. As mentioned in the method section, after preforming network influence iterations, we clustered the final output as a bi-partite graph with a method that does not require the specification of the number of clusters (Bi-Louvain [38]). As such, the method will obtain clusters of both the malware samples and the original clusters found from each mode of the malware data. The method may also have a different number of clusters than number of threat actor groups. The following table, Table 3 displays the clusters found by the CMIC method with their membership of malware samples and modal clusters.

The CMIC with a Bi-Louvain clustering produces roughly double the number of clusters as malware family labels. The found clusters also display heterogeneity in terms of size of the clusters with the largest cluster (CMIC cluster 1 on Table 3) having six times as many malware samples and modal clusters as the smallest cluster (CMIC cluster 6 on Table 3). The differences in cluster sizes are also the same for each of the cluster constituents; the largest cluster in terms of malware samples is also the largest cluster in terms of modal clusters, and so on.

CMIC Cluster	Number of Malware Samples	Number of Modal Clusters
1	674	32
2	617	29
3	365	17
4	242	16
5	173	6
6	114	4

TABLE 3: Numbers of malware samples and clusters from each mode of the malware samples in each of the final, CMIC-derived clusters.

Since there is distinct heterogeneity in the CMIC-found clusters, we then analyzed the relationship between the found clusters and threat actor groups. The following figure, Figure 3, displays the relative number of each threat actor found in each found cluster

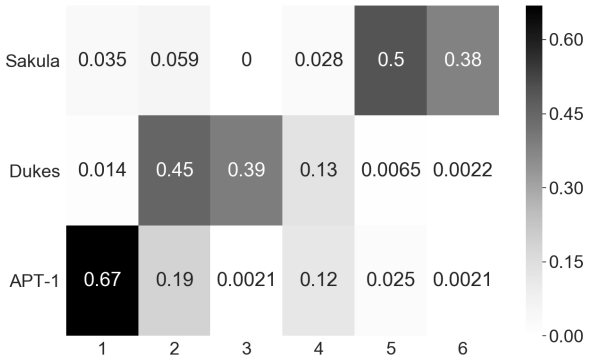


FIGURE 3: Fraction of each malware threat actor found in those clusters obtained by CMIC method.

The majority of APT-1 malware falls into the first, and largest cluster. Since APT-1 makes up most of the samples across all of the threat actors, it would stand to reason that it would dominant the larger clusters. More interestingly, the majority of samples between the Dukes and Depp Panda threat actors break into two clusters. In the case of Deep Panda, previous research on characterizing communities from malware samples found a similar result of their being actually two main communities of malware present in the Deep Panda threat group [5]. Thus, it may be possible that the Dukes and Deep Panda threat actors are more heterogeneous than the APT-1 in terms of their malware usage. This is an unusual result, since APT-1 and Dukes contain heterogeneous types of malware in its samples, while the Deep Panda samples are all from the Sakula family. This result may suggest that even though threat actors may use different tools, they may also share a lot of code between those tools, resulting in greater or lesser homogeneity in their malware. Also, this result may suggest that the Dukes and Deep Panda could actually be more than one threat actor. Finally, there is an outlier cluster in cluster number four.

This cluster does not contain a predominance of any of the threat actors and may represent unusual malware samples or a different malware actor entirely.

Since the samples from the Dukes threat actor contains some malware family labels as identified in [41], we can analyze how the Dukes malware families distribute across the CMIC-discovered clusters. The following figure, Figure 4, displays the relative number of each the Dukes malware families found in each found cluster.

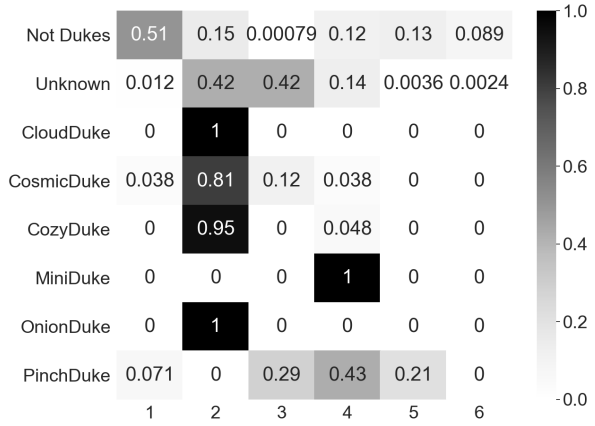


FIGURE 4: Fraction of Dukes malware families found in those clusters obtained by CMIC method.

Not unsurprisingly, most of the dukes samples fall within clusters 2 and 3 which contain most of the dukes samples. However, their distribution is not equal between clusters 2 and 3, with cluster 2 having most of the family-labeled Dukes malware samples. While its not clear why this division exists it may be related to when the samples were produced, with those in subgroup 3 coming from a later time period than those in subgroups 2, since the Dukes family labels come from a 2015 report on the Dukes threat actor [41] Furthermore, the one sample that belongs to the MiniDuke family fell into the outlier cluster (cluster 4) along with most of the PinchDuke samples. PinchDuke was an earlier family of malware used by Dukes which briefly had some code overlap with MiniDuke and was replaced by a near complete re-engineering of their malware tools [41]. So, this would suggest that subgroup 4 does indeed contain more of the anomalous samples across the different threat actors. Overall, it would seem given some of the family labels from the Dukes threat actor that the relationship between threat actors and the malware families that they employ can be a complex one.

Since the CMIC method also obtains clusters of the original modal clusters, we analyzed the relationship between the clusters of the different modes and the found clusters. The following figure, Figure 5, displays the fraction of each mode's clusters that are present in each of the found clusters.

Generally speaking, each mode's clusters are not equally distributed across all of the found clusters. The first two found clusters contain most of the clusters from all of the modes. This is not a surprising result, as the first two found

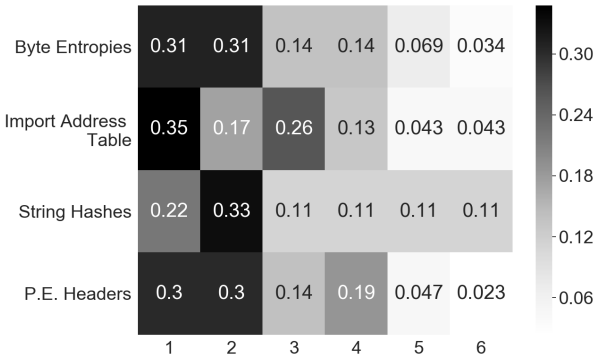


FIGURE 5: Fraction of each of the modes' clusters found in those clusters obtained by CMIC method.

clusters contain most of the malware samples across all of the modes. However, despite the preponderance of the first two found clusters, the distribution of modal clusters differs across the found clusters. For instance, both the clusters found from the import address table features and the clusters found from the string hashes have significantly different distributions of clusters across the found clusters than do those found using byte entropy and P.E. Header features. Given that found clusters also have different distributions of malware families, this result implies that there are differences in the modes of the features between the malware families. For example, cluster one contains mostly samples from the the APT-1 malware family whereas cluster two contains mostly samples from the Dukes malware family. These two found clusters also contain a lot of byte entropy and P.E. header clusters but differ significantly on the string hash clusters and import address table clusters. This may indicate that even though both of these malware threat actors use many different types of malware that could be similar in some respects, that their malware can still be distinct in other areas like their use of strings or import address table entries.

Overall, while the found clusters do generally follow the the threat actor groups there are some subtle differences between the two labelings as well. For two of the three threat actors, their found clusters actually tend to break into two main clusters which could indicate some distinct heterogeneity in those threat actors. Also, not all of the clusters found in each of the modes distribute the same way across all of the found clusters. This difference indicates that there are differences between the threat actor groups within the modes as well as across the modes.

E. ANALYSIS OF MULTI-MODAL NATURE OF MALWARE

One of the main findings in the previous section was that clustering the malware samples using intermediate and late integration, multi-modal approaches always produced better clusters relative to the malware threat actor labels than representing the malware as uni-modal through an early

integration approach. To investigate why this might be, we first investigated how well each mode's clusters relate to the threat actors. The following table, Table 4 presents the clustering goodness scores for each of the modal clusters, as found in the previous analysis by the mod-kNN procedure, relative to the threat actors.

Mode	ARI	AMI	Silhouette
Import Address Table	0.19	0.30	-0.85
Byte Entropies	0.17	0.26	-0.57
String Hashes	0.16	0.21	-0.57
P.E. Headers	0.13	0.22	0.19

TABLE 4: Cluster goodness results for each of the Berlin and Saxe feature sets, or modes, of the malware data.

For each mode of the malware data, no particular mode is especially good at clustering relative to the threat actors, however, the Import Address Table features do perform as good as the early integration clustering results. Thus, the Import Address Table features seem to have more relation to the threat actors than do any of the other sets of features. Furthermore, by concatenating all of the modes together in an early integration approach, the particular patterns of the Import Address Table become muddled with all of the other modes' patterns. In order to investigate this idea, we then analyzed how similar the clusters from each mode are to every other mode. The following set of figures, Figure 6, display the ARI and AMI values for each set of modes.

From the ARI and AMI values between the modal clusters, it is clear there is a fair amount of overlap between the cluster assignments of each of the modes. Most of the modes have ARI values greater than 0.4 and AMI values greater than 0.5 with every other mode, with the Byte Entropy features being the only exception. These high values indicate that the clusters for the malware samples between each of the modes are reasonably similar. However, the cluster assignments are not identical. This would indicate that using intermediate or late integration approaches for malware allows for exploitation of the differences in cluster assignments between the modes in order to produce better, overall cluster assignments. So, while not all modes of the Berlin and Saxe malware features are as useful as the others in identifying the possible threat actors from their malware usage, analyzing the data with intermediate and late integration approaches allows for better threat actor characterization because these methods can better exploit the differences in the clusters between the modes.

F. ANALYSIS OF USER-SET PARAMETERS OF CMIC

We now turn to analyzing how robust our proposed method is to changes in the user set parameters of the number of iterations and network influence strength, α. Beginning with the number of iterations to run the CMIC algorithm, we held the network influence strength value constant at $\alpha = 0.9$ and ran the algorithm for a number of iterations and recorded the AMI and ARI values at that iteration. The following plot, Figure 7 displays the ARI and AMI values as a function of the number of iterations of the algorithm.

As can be seen from the plot, the algorithm gets within good values of AMI and ARI in only 10 iterations. Furthermore, the algorithm stabilizes to its best AMI and ARI values by around 30 iterations. Thus, the algorithm not only converges to reasonable cluster labels quickly, but is also stable with the number of iterations.

Turning to the network influence parameter, α, we ran the CMIC algorithm for various values of α and fixed the number of iterations at 30. Most diffusion based processes, to include the social influence one we use in this work, typically have higher values of network influence ($\alpha >= 0.7$) [47], [48], [33]. So, we varied α from 0.5 to 0.99. The following plot, Figure 8, displays the goodness of clustering results for the various α values.

From the figure, when α is within normal ranges of 0.75 to 0.95, the resulting AMI and ARI values remain stable with less than a 0.02 variation in their values. Thus, the CMIC algorithm is also robust to the selection of the network influence strength parameter, α, as long as the value is within a reasonable range of 0.75 to 0.95. These results show that while a user must supply some parameters for our proposed algorithm, the algorithm is robust to the user input and can converge to reasonable cluster results quickly.

V. CONCLUSIONS AND FUTURE WORK

Our work presents the discovery and characterization of heterogeneous malware samples into threat actor groups from a multi-modal, unsupervised learning context. In our study we used the static malware features originally proposed by Berlin and Saxe in [4] to identify families of malware from three different families: APT-1, Deep Panda, and Dukes. Our results first demonstrated that finding communities in malware samples using intermediate and late integration, multi-modal approaches, instead of early integration approaches, always resulted in better threat group characterization. We believe this finding is a result of malware development more generally. Since many makers of malware share aspects of code between their different malware tools or only make small tweaks to a particular module of the malware to evade anti-malware products, then it should follow that having features and algorithms that are sensitive to the changes in only certain aspects of a malware sample should produce better results. So, a key finding of our research is that the characterization of malware samples into threat groups should be done from a late or intermediate integration, multi-modal approach.

Secondly, our proposed algorithm outperforms all the other algorithms in placing malware samples into their correct threat actor groups with our set of malware samples from three different threat actors. Our proposed method bridges the gap between the two lines of methodological research in multi-modal clustering by using cluster labels specific to each mode, but also uses diffusion across graphs constructed from each mode. Thus, by using advances from both lines of research we were able to design an algorithm that produces

(a) ARI (b) AMI

FIGURE 6: Comparison of the clusters found within the different modes of the features.

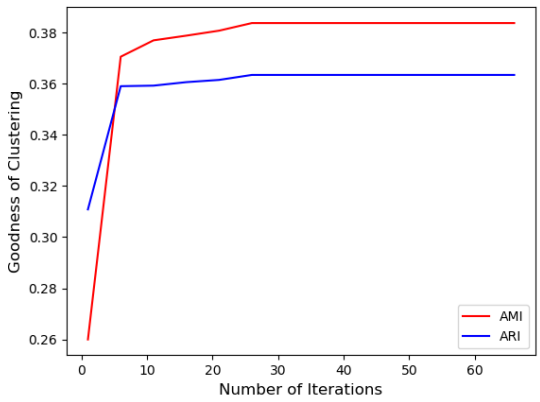


FIGURE 7: Iterations of the CMIC method versus the clustering goodness relative to the malware family labels

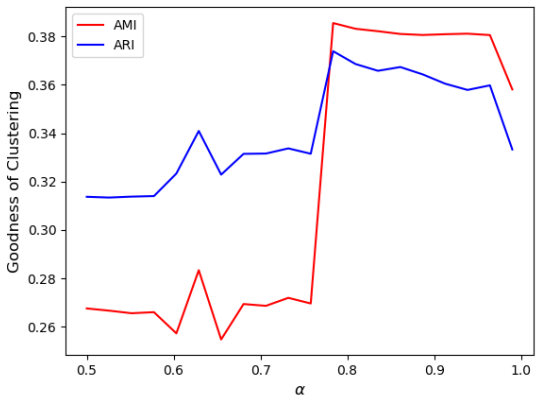


FIGURE 8: α values of the CMIC method versus the clustering goodness relative to the malware family labels

superior results in finding clusters of malware from multi-modal features.

The results of our investigation present several interesting avenues for future research. First, In our study we used a select set of static features derived from malware samples. It is well known that static features for malware identification can suffer from various forms of obfuscation applied to the malware [13]. It would be interesting to see if more and wider varieties of features taken from the malware samples, to include features from dynamic analyses, may be even better at identifying malware threat groups from a multi-modal data perspective. Second, it would also be interesting to investigate a wider variety of malware threat actor groups of different types of samples of malware (i.e. Trojans versus viruses, etc.) to see if the the multi-modal clustering approach still recovers useful groups of threat actors. In fact, while our experiments did produce reasonable threat actor groups in an unsupervised way, no particular algorithm performed well at finding the threat actor labels from the malware samples overall. It would seem that characterizing threat actors, and not just malware families, from malware samples remains a challenging problem. Finally, we also intend to investigate variations on the CMIC algorithm to include network influence thresholds and the use of weighted averaging during the mode-combining step.

REFERENCES

[1] Symantec, "Advanced persistent threats: A symantec perspective," tech. rep., 2012.

[2] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in Communications and Multimedia Security (B. De Decker and A. Zúquete, eds.), (Berlin, Heidelberg), pp. 63–72, Springer Berlin Heidelberg, 2014.

[3] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," ACM Comput. Surv., vol. 50, pp. 41:1–41:40, June 2017.

[4] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," pp. 11–20, 10 2015.

[5] I. J. Cruickshank, A. Johnson, T. Davison, M. Elder, and K. M. Carley, "Detecting malware communities using socio-cultural cognitive mapping," in 2019 International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation, 7 2019.

[6] T. Baltrusaitis, C. Ahuja, and L. Morency, "Multimodal machine learning: A survey and taxonomy," CoRR, vol. abs/1705.09406, 2017.

[7] S. Huang, K. Chaudhary, and L. X. Garmire, "More is better: Recent

progress in multi-omics data integration methods," Frontiers in Genetics, vol. 8, p. 84, 2017.
[8] M. Zhang, "Weighted Clustering Ensemble: A Review," arXiv e-prints, p. arXiv:1910.02433, Oct 2019.
[9] T. Boongoen and N. Iam-On, "Cluster ensembles: A survey of approaches with recent extensions and applications," Computer Science Review, vol. 28, pp. 1 – 25, 2018.
[10] H. Liu, Z. Tao, and Z. Ding, "Consensus Clustering: An Embedding Perspective, Extension and Beyond," arXiv e-prints, p. arXiv:1906.00120, May 2019.
[11] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," J. Mach. Learn. Res., vol. 3, pp. 583–617, Mar. 2003.
[12] Y. Ye, T. Li, Y. Chen, and Q. Jiang, "Automatic malware categorization using cluster ensemble," in Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, (New York, NY, USA), pp. 95–104, ACM, 2010.
[13] X. Hu and K. G. Shin, "Duet: Integration of dynamic and static analyses for malware clustering with cluster ensembles," in Proceedings of the 29th Annual Computer Security Applications Conference, ACSAC '13, (New York, NY, USA), pp. 79–88, ACM, 2013.
[14] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "From ensemble clustering to multi-view clustering," in Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, pp. 2843–2849, AAAI Press, 2017.
[15] A. Verma and K. K. Bharadwaj, "Identifying community structure in a multi-relational network employing non-negative tensor factorization and ga k-means clustering," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 7, 2 2017.
[16] L. Tang and H. Liu, Community Detection and Mining in Social Media. Morgan & Claypool Publishers, 2010.
[17] S. Bai, Z. Zhou, J. Wang, X. Bai, L. J. Latecki, and Q. Tian, "Automatic ensemble diffusion for 3d shape and image retrieval," IEEE Transactions on Image Processing, vol. 28, pp. 88–101, Jan 2019.
[18] A. Aleta and Y. Moreno, "Multilayer Networks in a Nutshell," Annual Review of Condensed Matter Physics, vol. 10, pp. 45–62, Mar 2019.
[19] B. Wang, J. Jiang, W. Wang, Z. Zhou, and Z. Tu, "Unsupervised metric fusion by cross diffusion," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2997–3004, June 2012.
[20] B. Wang, A. M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and A. Goldenberg, "Similarity network fusion for aggregating data types on a genomic scale," Nature Methods, vol. 11, pp. 333–337, 2014.
[21] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in AAAI, 2014.
[22] Y. Zhang, W. Yang, B. Liu, G. Ke, Y. Pan, and J. Yin, "Multi-view spectral clustering via tensor-svd decomposition," in 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 493–497, Nov 2017.
[23] J. Wu, Z. Lin, and H. Zha, "Essential tensor learning for multi-view spectral clustering," CoRR, vol. abs/1807.03602, 2018.
[24] Y. Liang, D. Huang, and C.-D. Wang, "Consistency meets inconsistency: A unified graph learning framework for multi-view clustering," in 2019 IEEE International Conference on Data Mining (ICDM), 11 2019.
[25] G. M. Campedelli, I. Cruickshank, and K. M. Carley, "Detecting Latent Terrorist Communities Testing a Gower's Similarity-Based Clustering Algorithm for Multi-partite Networks," in Complex Networks and Their Applications VII (L. M. Aiello, C. Cherifi, H. Cherifi, R. Lambiotte, P. Lió, and L. M. Rocha, eds.), vol. 812, pp. 292–303, Cham: Springer International Publishing, 2019.
[26] G. M. Campedelli, I. Cruickshank, and K. M. Carley, "A complex networks approach to find latent clusters of terrorist groups," Applied Network Science, vol. 4, p. 59, Aug 2019.
[27] H. Yu, Y. Lian, L. Zong, and L. Tian, "Self-paced learning based multi-view spectral clustering," in 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 6–10, Nov 2017.
[28] V. Gligorijević, Y. Panagakis, and S. Zafeiriou, "Non-negative matrix factorizations for multiplex network analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, pp. 928–940, April 2019.
[29] I. Brugere, B. Gallagher, and T. Y. Berger-Wolf, "Network structure inference, a survey: Motivations, methods, and applications," ACM Comput. Surv., vol. 51, pp. 24:1–24:39, Apr. 2018.
[30] J. Ruan, "A fully automated method for discovering community structures in high dimensional data," in 2009 Ninth IEEE International Conference on Data Mining, pp. 968–973, Dec 2009.
[31] M. E. J. Newman, Networks: An Introduction. Oxford University Press, 2010.
[32] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, p. 10008, Oct 2008.
[33] N. E. Friedkin and E. C. Johnsen, Social Influence Network Theory: A Sociological Examination of Small Group Dynamics. Structural Analysis in the Social Sciences, Cambridge University Press, 2011.
[34] A. Flache, M. Mäs, T. Feliciani, E. Chattoe-Brown, G. Deffuant, S. Huet, and J. Lorenz, "Models of social influence: Towards the next frontiers," Journal of Artificial Societies and Social Simulation, vol. 20, no. 4, p. 2, 2017.
[35] M. Donoser and H. Bischof, "Diffusion processes for retrieval revisited," in 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1320–1327, June 2013.
[36] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, (New York, NY, USA), pp. 36–, ACM, 2004.
[37] D. Huang, C. Wang, and J. Lai, "Locally weighted ensemble clustering," IEEE Transactions on Cybernetics, vol. 48, pp. 1460–1473, May 2018.
[38] C. Zhou, L. Feng, and Q. Zhao, "A novel community detection method in bipartite networks," Physica A: Statistical Mechanics and its Applications, vol. 492, pp. 1679 – 1693, 2018.
[39] D. Alperovitch, "Deep in thought: Chinese targeting of national security think tanks," tech. rep., 2014.
[40] Dell, "Sakula malware family threat analysis," Jul 2015.
[41] F-Secure Labs, "The dukes: 7 years of russian cyberespionage," tech. rep., 09 2015.
[42] D. Huang, J.-H. Lai, and C.-D. Wang, "Combining multiple clusterings via crowd agreement estimation and multi-granularity link analysis," Neurocomputing, vol. 170, pp. 240 – 250, 2015. Advances on Biological Rhythmic Pattern Generation: Experiments, Algorithms and Applications Selected Papers from the 2013 International Conference on Intelligence Science and Big Data Engineering (IScIDE 2013) Computational Energy Management in Smart Grids.
[43] J. R. Bray and J. T. Curtis, "An ordination of the upland forest communities of southern wisconsin," Ecological Monographs, vol. 27, no. 4, pp. 325–349, 1957.
[44] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," J. Mach. Learn. Res., vol. 11, pp. 2837–2854, Dec. 2010.
[45] L. Hubert and P. Arabie, "Comparing partitions," Journal of Classification, vol. 2, pp. 193–218, Dec 1985.
[46] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," Journal of Computational and Applied Mathematics, vol. 20, pp. 53 – 65, 1987.
[47] X. Yang, L. Prasad, and L. J. Latecki, "Affinity learning with diffusion on tensor product graph," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, pp. 28–38, Jan 2013.
[48] B. Wang, A. Pourshafeie, M. Zitnik, J. Zhu, C. D. Bustamante, S. Batzoglou, and J. Leskovec, "Network enhancement as a general method to denoise weighted biological networks," Nature Communications, vol. 9, p. 3108, Aug 2018.

IAIN J. CRUICKSHANK Iain J. Cruickshank is a Ph.D. candidate in Societal Computing at Carnegie Mellon University, as a National Science Foundation Graduate Research Fellow. He received a B.S. in mathematics from the United States Military Academy in 2010 and an M.S. in Operations Research from the University of Edinburgh, Scotland, in 2011 as a Rotary Ambassadorial Scholar.

He is a Captain in the United States Army with assignments that include being a National Mission Team Sub-Element Lead in the Cyber National Mission Force and a Company Commander for 781st Military Intelligence Battalion. His research interests include data science, machine learning, and the use of multi-modal data in understanding social phenomenon.

KATHLEEN M. CARLEY Kathleen M. Carley, Harvard Ph.D. in Sociology, H.D. University of Zurich, is a Professor of Societal Computing in the School of Computer Science at Carnegie Mellon University, IEEE Fellow, director of the center for Computational Analysis of Social and Organizational Systems (CASOS), and CEO of Netanomics. In 2019 she received an honorary degree from the University of Zurich in Business, Economics and Informatics. She is the 2011 winner of the Simmel Award from INSNA, and the 2018 winner of the USGA Academic Award, from GEOINT. She funded the area of social cybersecurity and leads an international group in that area. Research areas include social network analysis, dynamic network analysis, big-data network analytics, agent-based modeling, adaptation and evolution, social-network text-mining, information warfare, disinformation, cyber-security, information diffusion, social media and telecommunication, disease contagion, and disaster response. She and members of her teams have developed novel tools and technologies for rapidly extracting networks from texts and assessing sentiment (AutoMap, NetMapper) analyzing large scale geo-temporal multi-dimensional networks (ORA-Pro, ORA-Lite, ORA-WEB), identifying Bots and Memes, and agent-based simulations (Construct).