UNIVERSITY OF CALIFORNIA
RIVERSIDE

Methods for Mining Important User-Generated Contents and Behaviors From Online Platforms

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Risul Islam

December 2021

Dissertation Committee:

Dr. Michalis Faloutsos, Chairperson
Dr. Vagelis Papalexakis
Dr. Vagelis Hristidis
Dr. Kevin M Esterling

The Dissertation of Risul Islam is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, Prof. Michalis Faloutsos, who is an excellent educator, a brilliant researcher, and most importantly, a genuine human being. He has been patient with me during my whole Ph.D. life. He supported me in my tough times, allowed me space to grow, encouraged me to learn, and directed me to be successful in research. For someone like me who was alien to the environment after coming to a foreign land, Prof. Faloutsos has been a figure of trust and reliability from day one till the very end. I will carry his teachings with me not only as a researcher but also as a human being, for the rest of my life and will try to improve myself.

I'd also like to thank all my committee members, Prof. Vagelis Papalexakis, Prof. Vagelis Hristidis, and Prof. Kevin M Esterling for their time and valuable suggestion in my research which ultimately helped me complete my Ph.D. works.

I am very much grateful to my family. Whenever I felt weak, they were there for me. Their endless support and belief always encouraged me and helped me go through this long journey. I will always be indebted to my mother Rozy Afroza, and my father Mostofa Kamal for their unconditional and unlimited support. They have always been my anchor and without them I am nothing.

I can not forget my close friends in Riverside who were there for me. They have gone out of their way to drag me into their life, accepted me as their friend and for that, I will always be grateful to them. Especially, Md Omar Faruk Rokon, Nishat Ara, C M Sabbir Ahmed, Jawad Bappy, and Rimi Apu- you guys have supported me and loved me beyond my imagination. I will cherish your friendship for the rest of my life.

I thank all my cricket buddies with whom I become very good friends in such a short period of time. You guys are my only escape during this Ph.D. life. I enjoyed every moment I spent with you. I am forever grateful for your friendship and your company.

Finally, my labmates, Dr. Joobin Gharibshah, Dr. Darki, Shaila, Masud, Ben, Jakapun, and Arman. Thank you for all your support and help. I could not have asked for any better lab-mates and friends than you. You have accepted me and valued me more than I deserve. Thank you very much and I hope we all will stay connected just like we are now.

For everyone else I haven't mention, and/or I can not mention, thank you, everyone. This journey was never easy and it is impossible to walk alone in this path. I stand where I am because of all of you. I am grateful and I thank you. May God bless all of you.

To my parents, Mostofa Kamal and Rozy Afroza, for all the support.

ABSTRACT OF THE DISSERTATION

Methods for Mining Important User-Generated Contents and Behaviors From Online Platforms

by

Risul Islam

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2021
Dr. Michalis Faloutsos, Chairperson

How effectively can we extract useful information from online platforms? Our work is motivated by the observation that online platforms, such as security forums, gaming forums, and software archives, hide significant and useful information. We argue that mining this information can greatly benefit security analysts as it can reveal trends, patterns of behavior, emerging threats, and even malicious actors. This thesis spans three interrelated problems in this space. First, we address the problem of how we can identify interesting activities in a forum for which we have no prior knowledge. We develop a systematic tensor-based tool to identify "events", defined in a three-dimensional space of users, threads, and time. A key novelty is that we let the forum "reveal" the events of interest in an unsupervised manner, while we empower the tech-savvy end-users to easily tune some parameters to influence their focus if so desired. Second, we propose a novel method to expand the tensor decomposition approach to reveal a hierarchical structure from the multi-modal data in a self-adaptive way. So far, current tensor decomposition-based algorithms extract a flat clustering from the multi-modal data. We apply our hierarchical method on real data from six online forums,

which leads us to many interesting findings which validate the value of our approach. Third, we turn our attention to software archives that seem to harbor significant hacker activity with thousands of publicly available malware repositories. The goal is to understand the collaboration dynamics of the hackers and follow their footprints across forums as well. In our thesis, we use the data from four security forums, one gaming forum, and one software archive with 50K users, 60K threads, 150K posts, 8.5K repositories spanning over 5 years. We show that our approaches are powerful, as they are able to identify: (a) interesting communities of users, (b) meaningful hierarchies of communities and events, and (c) several tight-knit groups of hackers that collaborate on malware projects. For example, we identify some real events like ransomware outbreaks (55 users, 86 threads, December 2015, February 2016), the emergence of a black-market of decryption tools (34 users, 12 threads, February 2016), and romance-enabled scamming (82 users, 172 threads, March 2018). To maximize the impact of our work, we intend to make our tools and our datasets publicly available as a tangible contribution to the research community. In conclusion, we believe that our approaches and tools constitute important steps towards automated capabilities for shifting through the wealth of information in online platforms efficiently and effectively.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"How can a 17 year old kid from Florida [2] be reportedly the mastermind behind the recent hacking of Twitter?" This question is part of the motivation behind this work.

The security community has a fairly limited understanding of malicious hackers and their interactions. As a result, security practitioners do not really know their "enemy". On the one hand, the hacker community is fairly wide encompassing curious teenagers, aspiring hackers, and professional criminals. On the other hand, the hackers are surprisingly bold in leaving a digital footprint, if one looks at the right places in the Internet. For example, there are various online forums, where hackers not only boast of their successes, but also share various hacking related information. We observe that online platforms, such as security forums, gaming forums, and software archives hide significant and useful information.

It turns out that security forums contain a wealth of information that currently remains unexplored. Online security forums have emerged as a platform where users generally

initiate a discussion about their security-related issues. These forums aggregate valuable information in an unstructured way and initial work argues for a wealth of useful information: emerging threats and attacks, promotion of hacking skills, and technical tutorials. Apart from security forums, popular and public software archives, such as GitHub harbor **malware authors**, who create publicly-accessible malware repositories [144]. Hackers tend to collaborate among themselves to develop functional hacking tools. Sometimes they share these tools in online forums as well. We argue that mining the important and impactful information from online forums and software archives can greatly benefit security analysts as it can reveal trends, patterns of malicious behavior, emerging threats, and even malicious actors.

Therefore, the question that we answer in this thesis is: "*How effectively can we extract useful information from online platforms?*" To answer the above-mentioned main question, we answer the following three sub-questions basically. (a) "*How can we identify major events of interest in a forum in an unsupervised fashion?*", (b) "*How can we expand the tensor decomposition to reveal a hierarchical structure of the multi-modal data in a self-adaptive way?*", and (c) "*How can we begin to understand the ecosystem of malicious hackers based on their online footprint?*". The input for all these sub-problems is the online platform data (forum data and GitHub repository data). The desired outputs are as follows: (a) the key events that capture the main activity in the forum and could be of interest to a security analyst. For example, a security analyst would want to identify outbreaks of attacks, the emergence of new technologies, groups of hackers with tight and focused interests, and underground black markets of hacking services. (b) identified clusters and

their potentially hierarchical structure present in the data. And (c) the influential hackers and their collaborative behaviors.

We face several challenges in this thesis. First, the data is unstructured because we are dealing with user-generated contents. There is a lot of "noise", lack of structure, and an abundance of informal and hastily written text. Second, we want to solve the sub-problems in an unsupervised way: we want the security forums "tell" us its events of interest. And, finally, evaluation is hard because of lack of ground truth.

There is limited work for the problems as defined above. We can group prior efforts into two main families: (s) studies on online forum, (b) studies on GitHub. In online forum related studies, we are not aware of any work that utilizes the power of tensor decomposition and proposes a tool to systematically identify the important events and profile them using NLP. Also, despite the vast literature on tensor decomposition, we are not aware of any work that fully explores the hierarchical tensor decomposition and discover the hierarchical patterns from online forums. Most of the previous works on online forums focus on identifying emerging topics and threats [52, 137]. Other efforts report malware activity, focusing on hacking events, and much less, if at all, on the ecosystem of hackers [149, 151]. In GitHub related studies, we are not aware of a study that systematically profiles the dynamics of the online hacker ecosystem, and especially one considering software archives. Most of the previous efforts on GitHub follow a software-centric view or study GitHub at large without focusing on malware [23, 22, 19, 146]. We elaborate on previous works later.

The key contributions in this thesis are as follows. First, we let the forum "reveal" the events of interest in an unsupervised manner, while we empower the tech-savvy

end-users to easily tune some parameters to influence their focus if so desired. Second, we propose a novel method to expand the tensor decomposition approach to reveal a hierarchical structure from the multi-modal data in a self-adaptive way. Third, we unveil the collaboration dynamics of the hackers in GitHub and follow their footprints across forums as well.

Our key results are summarized in the following points.

(a) **We identified a total of 68 interesting events from a total of 52 clusters.** We find that **83%** of our identified clusters revolve around interesting events and each cluster shows high intra-cluster thread similarity, as validated by experts, crowdsourcing, and other methods.

(b) **We extract meaningful "hierarchical" clusters corresponding to real life events.** For example, one cluster from a forum discusses about Android malware which is then further divided into Android Version 5 malware cluster and Android Version 7 malware cluster.

(c) **We identify a group of 1.7% of influential malicious authors and 30 professional hackers from GitHub.** They are responsible for: (i) generating influential repositories, and (ii) providing the social backbone of the malware community.

We develop a systematic suit of mining capabilities for studying user-generated contents in online platforms. Our tools are capable of identifying (a) influential hackers, (b) their alarming activities, (c) communities of collaborating hackers, and (d) their cross-platform interactions. Follow up research can expand on our work to develop preemptive security initiatives, such as: (a) monitoring hacker activity, (b) detecting emerging trends,

and (c) identifying particularly influential hackers towards safeguarding the Internet. We believe that our methods can be seen as a great foundation to mine the wealth of information that exist in online platforms.

Chapter 2

Dataset

Our work focuses on user-generated data from online forums and uses data from GitHub, the largest software archive with roughly 30 million public repositories. We briefly describe the dataset below.

2.1 Online forum data

We utilize data that we collect from four security forums: Wilders Security, Offensive Community, Hack This Site, Ethical Hackers and from one gaming forum: Multi-Player Game Hacking Cheats (MPGH) [129]. In these forums, users initiate discussion threads in which other interested users can post to share their opinion.

For completeness, we start with some terminology. Each *thread* has a *title* and is started by its first post, and we refer to subsequent posts as *comments*. The *duration* of a thread is defined by the time difference between the first and last post of that thread. The *active days* for a forum are the number of days when the dataset contains at least one post.

Each tuple in our dataset maintains the following format, F:=(forum ID, thread ID, post ID, username, date, and post content). We provide a brief description of our forums below, and an overview of key numbers in Table 2.1.

a. OffensiveCommunity (OC): As the name suggests, this forum contains "offensive security" related threads, namely, breaking into systems. Many posts consist of step by step instructions on how to compromise systems, and advertise hacking tools and services.

b. HackThisSite (HTS): As the name suggests, this forum has also an attacking orientation. There are threads that explain how to break into websites and systems, but there are also more general discussions on cyber-security.

c. EthicalHackers (EH): This forum seems to consist mostly of "white-hat" hackers, as its name suggests. However, there are many threads with malicious intentions in this forum.

d. WildersSecurity (WS): The threads in this forum fall in the grey area, discussing both "black-hat" and "white-hat" skills.

e. Multi-Player Game Hacking Cheats (MPGH): MPGH is one of the largest online gaming communities with millions of discussions regarding different insider tricks, cheats, strategy, and group formation for different online games. The dataset was collected for 2018 and contains 100K comments of 37K users [133].

Forum	Users	Threads	Posts
Offensive Comm.	5412	3214	23918
Ethical Hacker	5482	3290	22434
Hack This Site	2970	2740	20116
Wilders Security	3343	3741	15121
MPGH	37001	49343	100007

Table 2.1: Statistics of our five online forums.

2.2 GitHub data

GitHub platform enables software developers to create software repositories in order to store, share, and collaborate on projects and provides many social-network-type functions.

We define some basic terminology here. We use the term *author* to describe a GitHub user who has created at least one repository. A *malware repository* contains malicious software and a *malware author* owns at least one such repository. Users can *star, watch* and *fork* other *malware repositories. Forking* means creating a clone of another repository. A forked repository is sometimes merged back with the original parent repository, and we call this a *contribution.* Users can also *comment* by providing suggestions and feedback to other authors' repositories.

We use a dataset of 7389 malware authors and their related 8644 malware repositories, which were identified among 97K repositories in our prior work [144]. This is arguably the largest malware archive of its kind with repositories spanning roughly 11 years. These repositories have been identified as malicious with a very high precision (89%). Note that the queries with the GitHub API, which were used in the data collection, return primary or non-forked repositories. A discussion on the process, accuracy, and validity of the dataset can be found in the original study [144].

For each malware author in our dataset, we have the following information: (a) the list of the malware repositories created by her, and (b) the list of followers. For each malware repository, we have the lists of users, who: (a) star, (b) watch, (c) fork, (d) comment, or (e) contribute to the repository.

Repository metadata. Each repository is also associated with a set of user generated fields, such as title, readme file, description. We can use this *metadata* to extract information about the repository. We leverage our earlier work where we discuss the processing of this metadata in more detail [144].

For a given repository, a security expert would want to know: (a) the type of malware (e.g. ransomware and keylogger), and (b) the target platform (e.g. Linux and Windows). For this, we define two sets of keywords: (a) 13 types of malware, S_1 and (b) 6 types of target platforms, and S_2. Fig. 5.6 provides a visual list of these two sets of keywords. We define the Repository Keyword Set, W_r, for repository r, as a set consisting of the keyword sets S_1 and S_2 that are present in its metadata. Clearly, one can extend and refine these keyword sets, to provide additional information, such as the programming language in use, which we will consider in the future. Note that our earlier work provides evidence that using this metadata as we do here can provide fairly accurate and useful information [144].

Chapter 3

TenFor: A Tensor-Based Tool to Extract Interesting Events from Security Forums

3.1 Introduction

Security forums contain a wealth of information that currently remains unexplored. Online security forums have emerged as a platform where users generally initiate a discussion about their security-related issues. These forums aggregate valuable information in an unstructured way and initial work argues for a wealth of useful information: emerging threats and attacks, promotion of hacking skills, and technical tutorials. Discussions around these topics at one or more points in time often involve a large number of users and threads, and we can think of them as important **events** in the life of the forum.

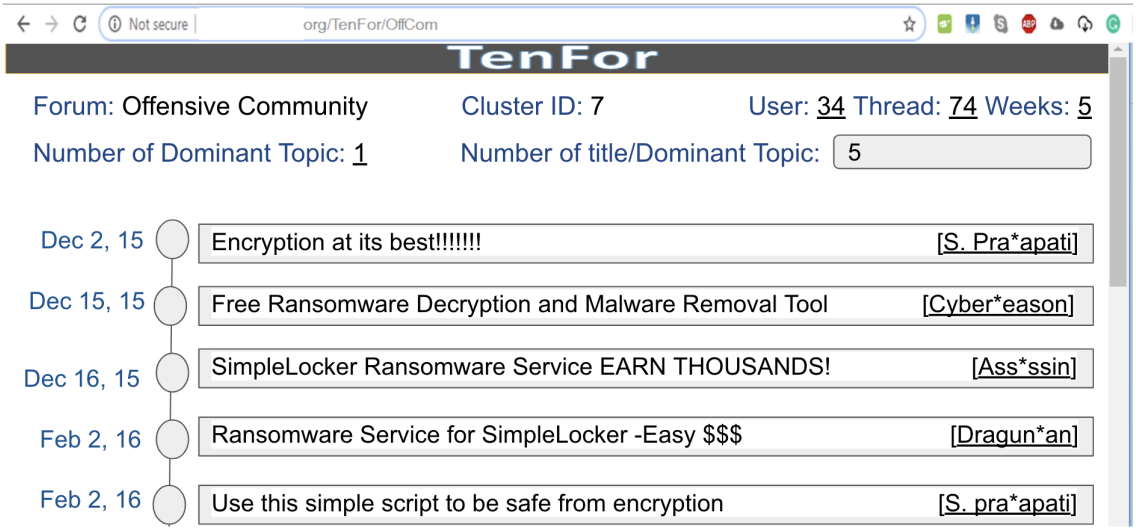


Figure 3.1: StoryLine View: a user-friendly visualization of a cluster summary with our TenFor tool. We present one of the identified clusters, which captures the emergence of SimpleLocker ransomware from the Offensive Community forum.

How can we identify major events of interest in a forum in an unsupervised fashion? The input is a forum, and the desired outputs are the key events that capture the main activity in the forum and could be of interest to a security analyst. For example, a security analyst would want to identify outbreaks of attacks, the emergence of new technologies, groups of hackers with tight and focused interests, and underground black markets of hacking services. The challenges are that the information is unstructured and that we want to do this in an unsupervised way: **we want the forum to tell us its events of interest**.

Mining security forums has received relatively little attention and only recently. We can identify three main categories of related efforts: (a) security forum studies, (b) analysis of blogs, social media, and other types of forums, and (c) tensor-based mining approaches. We discuss these efforts in Section 3.5.

As our key contribution, we propose, TenFor, a systematic tensor-based approach and tool to identify important events in an unsupervised way in a forum. Our approach operates at the three-dimensional space of (a) users, (b) threads, and (c) time. Our method consists of three main steps: (a) *clustering* using a tensor decomposition, (b) *profiling* using both content and behavioral metrics, and (c) *investigating* using an automated, but customizable, method to capture the dynamics of the cluster and provide an interpretable view.

We can summarize the novelty of our approach in terms of techniques and features as follows: (a) it operates in an unsupervised way, (b) it adapts and combines tensor-based clustering, behavioral profiling, and NLP methods, (c) it is user-friendly by being parameter-free with *optional* user-specified "knobs" that can adjust the granularity and information detail of the results, and (d) it provides visual and intuitive fingerprints of the events of interest. All these capabilities are further discussed later.

Overall, an end-user can obtain the following results: (a) the most dominant clusters in the lifespan of the forum, (b) profiling information about these clusters including key users, key threads, key dates, and key topics and keywords, (c) optional labeling of the clusters using user-specified keywords. Visually, the results can appear in a StoryLine View or a Table View as shown in Fig. 3.1 and Table 3.3 respectively.

In our evaluation, we apply TenFor on three security forums and one gaming forum with a total of 54000 users. We find that **83%** of our identified clusters revolve around interesting events and each cluster shows high intra-cluster thread similarity, as validated by experts, crowdsourcing, and other methods. Our approach also compares favorably with

previous approaches [155, 6] leveraging the power of tensor decomposition to strike the balance between size and number of clusters.

Going beyond security forums. Although we focus primarily on security forums here, TenFor can be used on other types of forums. As a proof of this, we apply our approach on an online gaming forum and find interesting activities, including revenge hacking and romance scamming, as we discuss later.

The overarching vision. As a tangible contribution, we develop a powerful user-friendly platform that will be useful to both researchers, and industry practitioners. Our ambition is to make this platform a reference tool for forum analysis and inspire subsequent research and development.[1] The proposed hands-free event extraction is a significant capability: we let the forum to tell us what are the key activities of interest, namely "taking the pulse" of the forum. This can enable practitioners to shift through a large number of forums of interest efficiently and effectively. In the future, we will extend our tool by providing additional user-centric and content-centric capabilities.

3.2 Datasets

We apply our method on four forums in our archive, which consists of three security forums: Offensive Community (OC), Ethical Hacker (EH), Hack This Site (HTS), and one online gaming forum: Multi-Player Game Hacking Cheats (MPGH) [154].

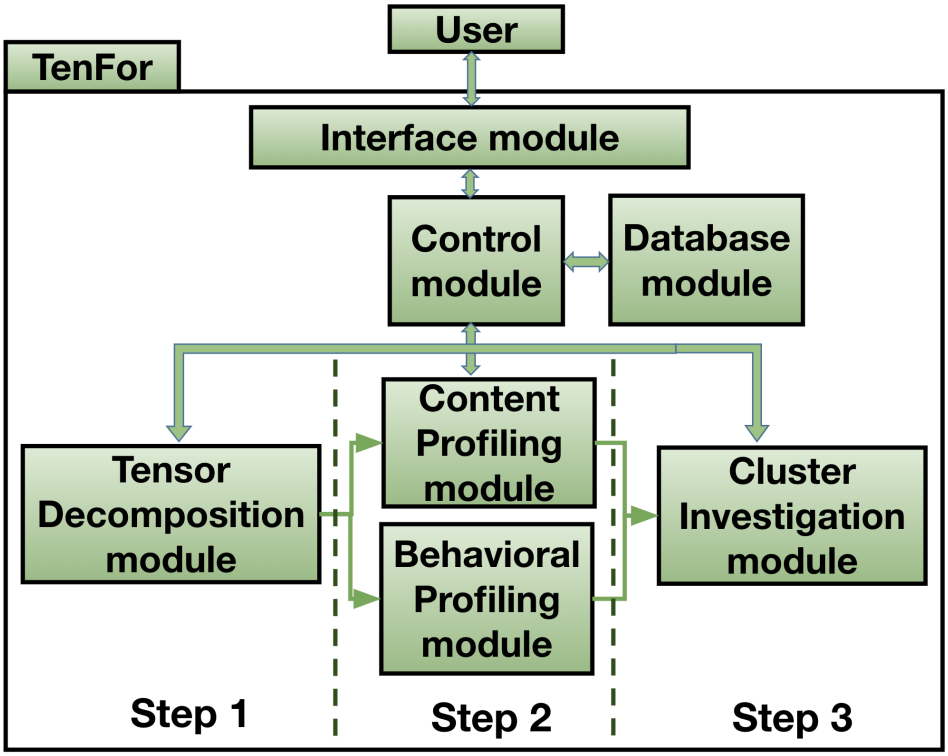


Figure 3.2: Overview of the TenFor approach and its steps: Step 1: Cluster; Step 2: Profile; Step 3: Investigate.

3.3 Our Approach

We present, TenFor, a tensor-based multi-step approach, that identifies events and activities in an unsupervised way. Fig. 3.2 provides the architecture of the platform. The Control module communicates with Interface and Database modules. The algorithmic core is provided by the Tensor Decomposition, Content Profiling, Behavioral Profiling, and Investigation modules.

We present an overview of TenFor, which works in 3 steps: a) clustering via tensor-based decomposition, b) cluster profiling, and c) cluster investigation as shown in Fig. 3.2.

[1]Sample code: https://github.com/RisulIslam/TenFor

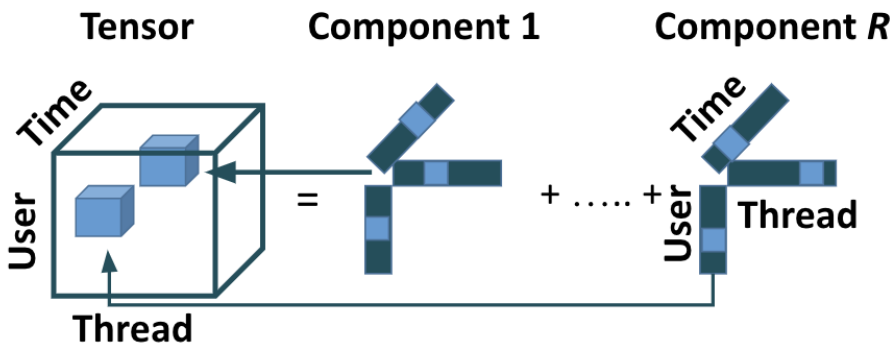


Figure 3.3: Visualization of tensor decomposition.

Automated operation with optional user control. A key design principle of our approach is to operate parameter-free, and at the same time, provide the end-users with "knobs" for obtaining results of interest. Naturally, a savvy end-user can exert even more control by specifying algorithmic parameters with well-defined APIs, especially in the tensor decomposition, which we discuss below. We revisit these parameters at the end of this section.

3.3.1 Step 1: Tensor-based clustering

We provide an overview of the challenges and algorithmic choices in our approach, starting with an introduction to tensors and tensor decomposition.

Tensors and decomposition. A d-mode tensor [91] is a d-way array (here we use $d = 3$). So, we call $I \times J \times K$ tensor a "3-mode" tensor where "modes" are the fixed number of indices to index the tensor; for us the "modes" being the user (U), thread (T), and weekly discretized time (W). Each 3D element of the tensor, *X(i,j,k)*, captures the total number of interaction (in terms of #comments) of user i in thread j at discretized week k or zero in

the absence of such interaction. In a decomposition, we decompose a tensor into R rank-one components, where R is the rank of the tensor, as shown in Fig. 3.3. That means tensor is factorized into a sum of rank-one tensors i.e. a sum of outer products of three vectors (for three modes): $X \approx \sum_{r=1}^{r=R} U(:,r) \circ T(:,r) \circ W(:,r)$ where $U \in \mathbf{R}^{I \times R}$, $T \in \mathbf{R}^{J \times R}$, $W \in \mathbf{R}^{k \times R}$ and the outer product is derived by $(U(:,r) \circ T(:,r) \circ W(:,r))(i,j,k) = U(i,r)T(j,r)W(k,r)$ for all *i, j, k*. Each component represents a latent pattern in the data, and we refer to it as **cluster**. For example, one such cluster in OC represents a group of 29 users that are active in the first weekends of July 2016 and discuss "multi-factor authentication failure" in a group of 72 threads. Each cluster is defined by three vectors, one for each dimension, which show the participation strength of each element for that cluster. Typically, one considers a threshold to filter out elements that do not exhibit significant participation strength, as we discuss later.

We need to address the following challenges to make the tensor decomposition work well in our domain.

a. What is the ideal number of components to target in the decomposition? To answer this question, we use the AutoTen method [131] and find the rank (R) of the tensor, which points to the ideal number of clusters to be decomposed into. AutoTen uses the *Core Consistency Diagnostic* metric in `CP_ALS` and `CP_APR` to find two probable ranks and finally chooses the max rank for the decomposition. So, the final rank, R, of a tensor is computed as follows: $R = max(R_{\texttt{CP\_ALS}}, R_{\texttt{CP\_APR}})$.

b. How can we decompose the tensor? We use the Canonical Polyadic or CANDECOMP/ PARAFAC (CP) decomposition to find the clusters. The factorization may

contain negative numbers in the decomposed components whereas our strategy of capturing the interaction between users and threads at different times is inherently non-negative. We can achieve the non-negative factorization by adding the non-negative constraint in CP decomposition.

c. How can we strike a balance on cluster size? Each cluster is defined by three vectors (user, thread, and time), whose lengths are equal to the dimensions of the tensor as shown in Fig. 3.3. We need a threshold to determine significant participation in the cluster, which is a common practice for (a) avoiding unreasonably dense clusters [150], (b) enhancing interpretability, and (c) suppressing noise. So, the challenge is to impose this sparsity constraint and eliminate the need for ad-hoc thresholding to find the clusters with only significant users, threads, and times. Our solution is to add L_1 norm regularization with non-negative CP decomposition. L_1 regularization pushes the small non-zero values towards zero. Therefore, for each vector, we filter out the zero-valued elements and produce clusters with significant users, threads, and weeks only. In this way, we can eliminate the noisy users, threads, and weeks having the least significant contributions in the forum. The final model that we use for finding the clusters looks like this:

$\min_{U\geq 0,T\geq 0,W\geq 0} \|X - D\|_F^2 + \lambda(\sum_{i,r} |U(i,r)| + \sum_{j,r} |T(j,r)| + \sum_{k,r} |W(k,r)|)$ where λ is the sparsity regularizer penalty (set to 1) and $D = \sum_r U(:,r) \circ T(:,r) \circ W(:,r)$. To find the clusters, we solve the above equation. Since the equation is highly non-convex in nature, we use the well-established Alternating Least Squares (ALS) optimizer as the solver. An example of a cluster after filtering is shown in Fig. 3.4 and is further discussed in Section 3.4.

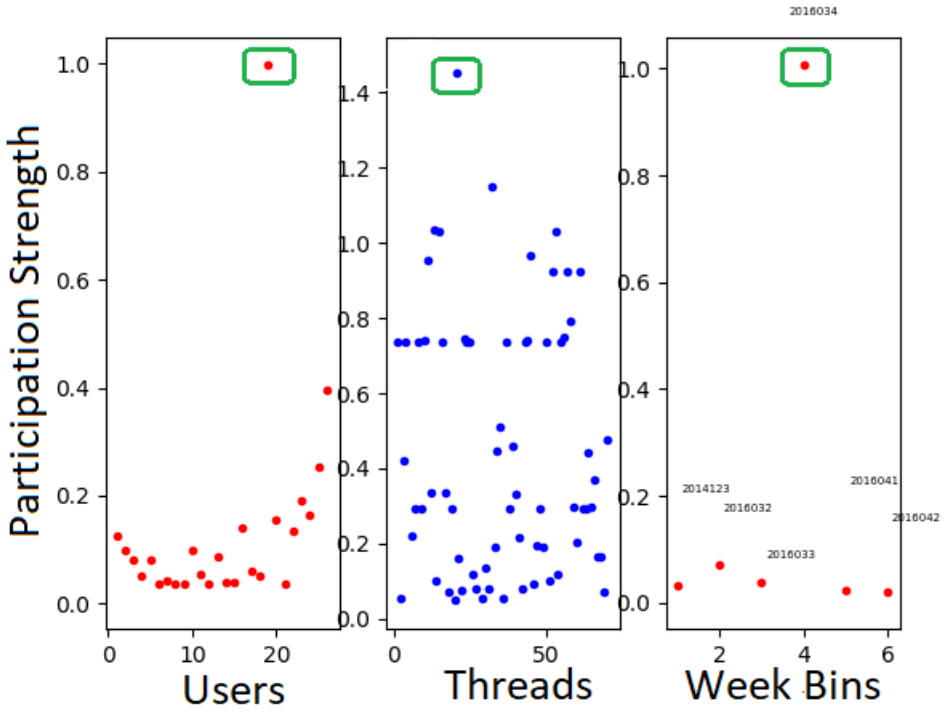


Figure 3.4: An example of a cluster (28 Users, 70 Threads, 6 Weeks) from OC. The intensity in each vector helps us identify users, threads and time intervals that are "important" for the cluster.

3.3.2 Step 2: Profiling the clusters

Having obtained the clusters, we propose to use content-based and behavior-based profiling to provide information and context for each cluster.

Step 2.1. Content-Based Profiling: We propose to profile clusters using content with the aid of two interconnected steps.

a. Cluster characterization: We identify the top N keywords using TF-IDF from the first post of each thread in each cluster. Prior work argues that the first post of a thread captures the focus of the thread [52]. We use the term *cluster keywords* to refer to

this set of words. These keywords can already provide a feel for the nature of the cluster, but we also use more sophisticated techniques in the next step.

b. Cluster labeling: We give the end-user the ability to define classes of interest that we then use to label the clusters. For ease of use, the end-users can define a class by providing a bag of words. To label the clusters, we compare these bags of words with the *cluster keywords* from the previous step.

To demonstrate this capability, we start with a group of classes that would be of interest to a security analyst. Specifically, we adopt the following classes of interest from prior work [52], which defines four types of threads: **Announcement type (A)** where people announce news and events, including hacking achievements and cyber-attacks; **Product type (P)**, where people buy or sell hacking services and tools; **Tutorial type (T),** where users post tutorials on how to secure or hack into systems; and, **General Discussion type (G)**, which is the category for all threads not in the above categories.

We then calculate how "relevant" each cluster is to each class type. For consistency, we have adopted the same definitions for these categories as the aforementioned work. To do this, we compute the Jaccard Similarity between the *cluster keywords* and the keywords that define each class type. We label the cluster as *A, T, P, G* type based on the highest Jaccard Similarity score. A cluster can be labeled as **Mix** type if the similarity scores of different types are within a close range (defined as ± 0.02).

Step 2.2. Behavior profiling: To provide more information per cluster, we use behavioral properties, which capture how users and threads interact with each other over time. We provide the following groups of capabilities and plots to the end-users:

a. Basic Distribution plots of metrics of the clusters in a forum, such as the distribution of #users, #threads, #active days etc. per cluster of the forum.

b. Scree-plots of metrics of clusters, which capture the pair-wise relationships of different metrics of clusters, such as #threads vs #users, % of active days vs duration (defined as the time difference between the last and the first post of the cluster) for each cluster of the forum as shown in Fig. 3.5 and 3.6.

c. Heat map visualizations of the clusters and the relative strength of their behavioral metrics. Currently, we use ten behavioral metrics that include the average (over the cluster): average post length per user, number of threads initiated per user, comment to thread participation ratio of the users, number of comments per user, number of active days of the threads etc. We normalize the values of the averaged metrics and present the behavioral profiles using a heat-map-style plot as we show, and discuss later, in Fig. 3.7.

The visual depiction helps an analyst to quickly gauge the behavioral profile of the clusters and spot differences. Also, we expand this functionality by developing an automated capability to report the anomalous cluster/s using standard DBScan anomaly detection algorithm [159] in these profiles. We discuss the findings in Section 3.4.

3.3.3 Step 3: Investigation of clusters

We develop a suite of capabilities that can help automate an in-depth investigation of the clusters coming from the previous steps. Although this can be done manually, the goal is to make the life of an analyst easier. Our platform provides the user with well-organized and easily accessible information trying to strike the balance between being informative and intuitively interpretable. Moreover, we develop two ways so that the end-users can

summarize the clusters: (i) StoryLine View, and (ii) Table View. These views help the end-users gauging through the clusters very quickly.

Step 3.1. Creating the StoryLine View: We develop a systematic and, arguably, more interpretable method to capture the essence of a cluster by highlighting the k most indicative threads in a non-decreasing temporal order as shown in Fig. 3.1. To accomplish this, we follow the process described below.

Identifying the important threads for the cluster is calculated in the following stages. In stage one, we find an extended list of topics, T_{ext}, for the whole cluster. To do this, we use the commonly-used LDA Bag-of-Words model [157], and we focus on the *titles* of the threads in the cluster threads because the *titles* provide a compact and meaningful summary of the threads. In stage two, we calculate the *relevance scores* of each thread with respect to each topic $t \in T_{ext}$. We associate each thread with the topic with the highest *relevance score*. In stage three, we find the most representing topics, T_{dom}, of the cluster. To achieve this, we find the distribution of the number of threads per topic in the decreasing order and from there we choose the list of *dominant topics,* T_{dom}, which we define as the minimum number of topics that represent at least "thread threshold", Th_{dom}=70%(default) of the threads. In stage four, we identify the top most relevant threads based on their relevance score for each of the dominant topics in T_{dom}. We then present them in a non-decreasing temporal order as shown in Fig. 3.1. Note that the parameter has a default value of 5, but the end-user can adjust it to her liking. Here, we focus on the *titles* as we want to have the title of thread "tell the story" in a visceral and intuitive way for the end-users. In the future, we will consider the text of the whole thread to find topics.

Step 3.2. Creating the Table View: We provide an alternative way to view all the clusters in the forum in a way that puts emphasis on key authors and key threads. This Table View can provide compact event summarization and key entities in each cluster. We argue that this may be appealing for a different type of analysis. Table 3.3 demonstrates the Table View that we provide. In our platform, we have clickable links that one can follow to investigate these entities of interest providing an interactive capability. We now present the generation of the columns of Table 3.3.

a. Identifying important entities: users, threads, and time intervals. We propose a method to identify the most dominant users, threads, and time periods, where significant activity takes place and populate the columns 4, 5, and 6 in Table 3.3. Specifically, we propose to identify the top k entities from each cluster, where $k \geq 1$ with the default being k=3. We use the factorized vectors to gauge the "importance" of an entity in a cluster as shown in Fig. 3.4. The green boxes show the entities with the highest "Participation Strength". From each of the top k weeks, we also report the most active day in terms of the highest #post made in that week.

Note that the parameter k can be modified by the end-user to adjust to her preference or type of investigation.

b. Representing the nature of the cluster in Table View: We present another way to capture the essence of a cluster, which we provide as text in the last column of Table 3.3. Obviously, there are many ways to achieve this. We opt to report the most dominant topics, T_{dom}, per cluster which is a common practice to represent and interpret events [160, 114]. We have already discussed a method to identify the dominant topics

Forum	Filtered Entities in clusters			# Cluster	# Cluster per Type				Type A (%)		Type T (%)		Type P (%)	
	U	Th	W		A	T	P	Mix/G	U	Th	U	Th	U	Th
OC	1086	2505	107	25	7	5	5	8	5.7	21.2	5.1	14.3	3.8	14.3
HTS	196	676	59	12	3	3	3	3	1.5	7.9	2	3.7	1.6	3.7
EH	315	424	82	15	3	6	2	4	1.1	2.3	2.8	2.2	0.6	1.3

Table 3.1: Properties of the clusters in OC, HTS and EH. Here U=user, Th=thread, W=weeks and the percentage of users, threads in a particular type of cluster is based on total number of users, threads in each forum.

above, which can be provided in the final column in Table 3.3. Note that in Table 3.3, we start from the dominant topics, but reconstruct the events within each cluster to provide more context to the readers.

The optionally tunable parameters of TenFor: TenFor can operate without any user input, but we expose the following parameters to a savvy user who wants to experiment, which we list here along with their default values: (a) temporal granularity: week, (b) size of the cluster keywords *N: 50*, (c) cluster labels: A, T, P, G/Mix as defined here, (d) thread threshold for dominant topic Th_{dom}=70%, (e) #relevant threads in StoryLine View R_t: 5, and (f) #top entities in Table View k: 3.

3.4 Results and Evaluation

We apply our method on four forums in our archive discussed in Section 3.2. We discuss the output of each step of TenFor for three security forums below.

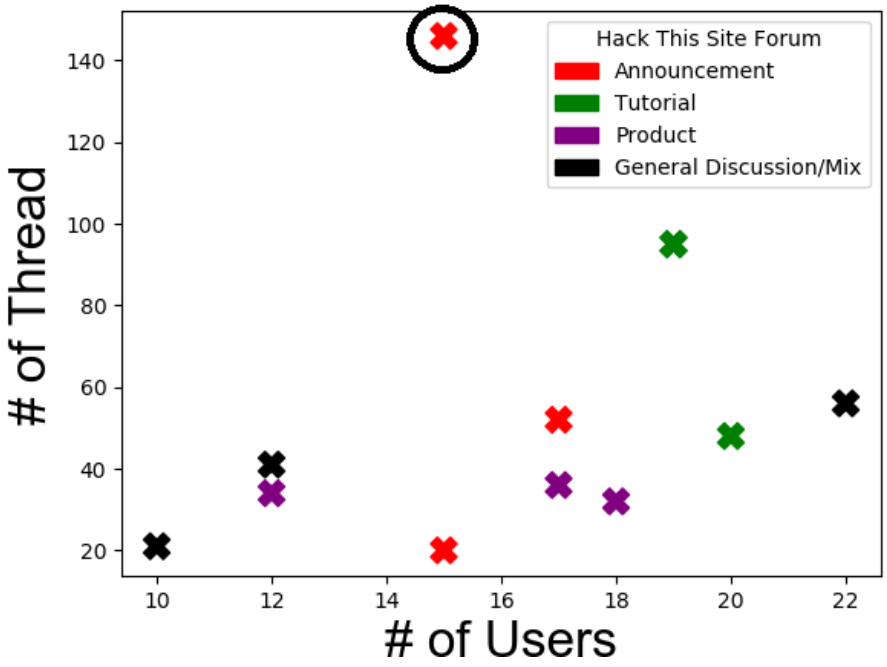


Figure 3.5: Scree plots of #users vs #threads in A, T, P, Mix/G type clusters of HTS.

3.4.1 Step by step output provided by TenFor

Step 1. We do a tensor decomposition for each forum. We provide an overview of the results of our decomposition in Table 3.1. Note that we opt to use *week* as the unit of time, but we experimented with days and months. Since we want to capture events, a week seems to strike a good balance between a day, and a month, which could be too short and too long respectively. We find the target number of clusters with the method described earlier. We get a total of 52 clusters from all three forums of which 25 clusters from OC, 12 clusters from HTS, and 15 clusters from EH. Note that we did experiment with more clusters than the ideal number, but that yielded extremely small clusters (e.g. 2 users, 3 threads, 1 week).

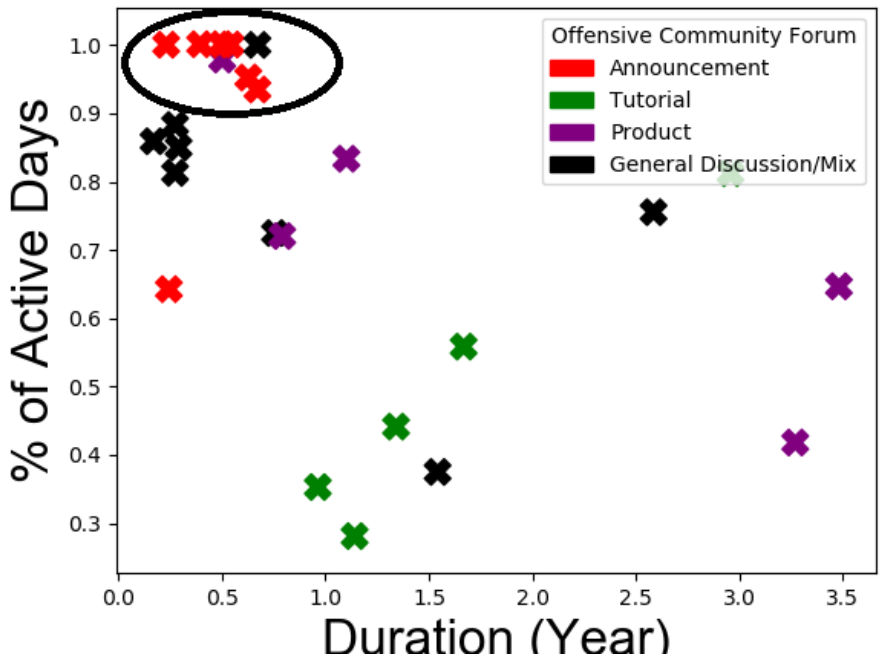


Figure 3.6: Scree plots of % of Active Days vs Duration in A, T, P, Mix/G type clusters of OC.

Step 2a. Content-based profiling and labeling. We use the A, T, P, or Mix/G labels, which we defined earlier. We set the # *cluster keywords, N*=50. Note that we report Mix and G types together here for the ease of presentation.

An overview of the clusters and their properties for all three forums is presented in Table 3.1. Specifically, we find the following distribution of clusters: (a) **26% of the clusters correspond to real security events**, such as attacks, (b) **22% of them represent black market communities** for malware tools and services, and (c) **32% of them represent security tutorials, events, and communities**, with most tutorials sharing malware and penetration techniques.

Step 2b. Behavior profiling. We provide the functionality to profile clusters based on their activity and dynamics. Apart from providing a general understanding, the

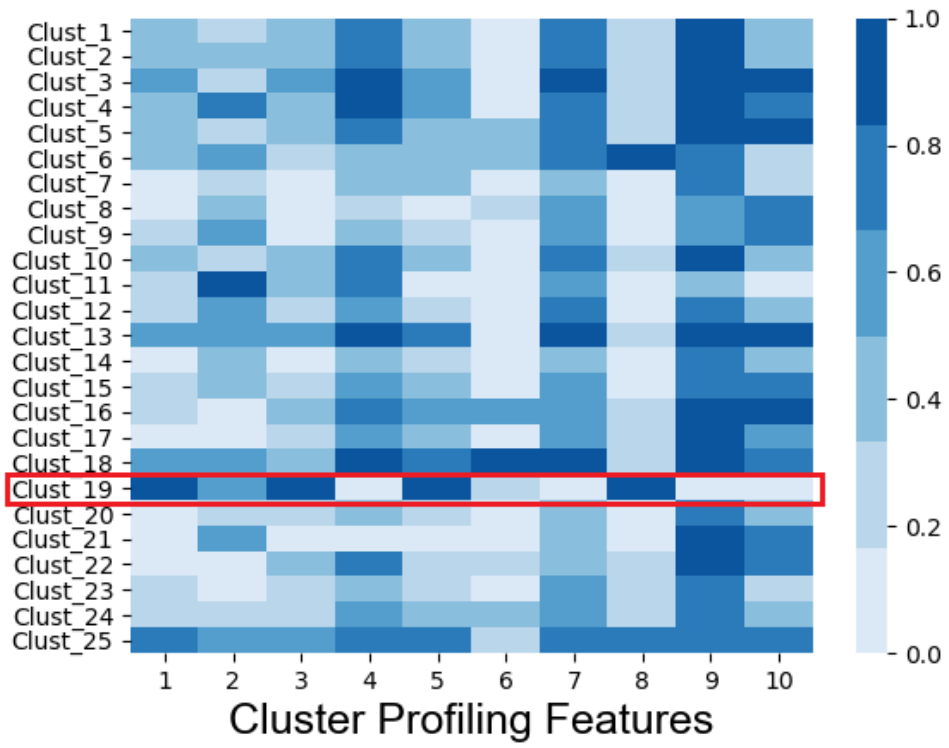


Figure 3.7: Behavioral profiling of the clusters from OC: x-axis is different behavioral features, and y-axis is cluster IDs. Case-study: Cluster 19 has a unique feature intensity profile.

analysis can help us spot outliers, which the end-users can investigate in Step 3 and have a summarized view of the clusters.

First, our TenFor platform provides some basic distribution plots, scree plots, and a heat map of the behavioral-based profile for each cluster, as described earlier. Due to space limitations, we only show two indicative scree plots in Fig. 3.5 and 3.6. In Fig. 3.5 for HTS, the black-circled cluster at the top is an **A** cluster, where just 15 people participate in a comparatively huge number of 145 threads. Upon further inspection, they are a group of hackers boasting about their hacking success. Some indicative *cluster keywords* of this cluster are *hack, brag, success, breach* etc.

Similarly, in Fig. 3.6 for the OC forum, the encircled clusters exhibit continuous activity: the Percentage of Active Days over the Duration of the cluster is more than 90%! This is an indication of an "urgency" in the cluster when compared with the typically lower Percentage of Active Days. This urgency is amply illustrated by cluster 9 (22 users, 60 threads): users talk about "strike week", during which the government attacked organized cyber-crime in the second half of March 2015. Strike week created frantic activity in the forum at that time.

Finally, we also provide a compact visual behavioral profile for each cluster shown in Fig. 3.7 for the OC forum. This can convey condensed information to the end-users visually. For example, cluster 19 (40 users, 88 threads), highlighted with the red box, seems to have a rare combination of active (dark blue) features. Specifically, these features suggest that the cluster exhibits high values of (a) average length of the first post of a thread per user (feature 1), (b) average ratio of #comments to #threads which a user generates or participates in (feature 3), and (c) average #comments per thread (feature 5). This behavior of the cluster is aligned with a *Tutorial* type cluster: (a) the first post is usually long, (b) tutorials often spark discussions, leading to multiple comments by a user in a thread, and (c) there are many questions and "thank you" comments in a tutorial thread. Note that this is also the label that our content-based labeling suggests.

Step 3. We showcase how we can enable a deeper analysis for each cluster with (a) Table View, and (b) StoryLine View. An example of our Table View is presented in Table 3.3 where we highlight three selected clusters from each forum and we provide the information in terms of the type of the cluster, most significant threads, users, and

dates. The final column is populated with the *dominant topics*, though here, we provide a manually-enhanced reconstruction of events for presentation purposes. As explained earlier, we also present a StoryLine View where we identify the top-k most indicative threads of the cluster which provides a human-readable thumbprint of the cluster. In Fig. 3.1, we show such a result that was generated automatically for cluster 7 (34 users, 125 threads) of the OC forum. We find that one topic, ransomware, represents 81% of the *titles*. In default settings, TenFor reports top *k=5 titles* based on the highest *relevance score* for the "ransomware" topic in a sorted timeline fashion. From this StoryLine View, the analyst can easily come into a conclusion that the cluster actually captures the spread of SimpleLocker. Therefore, this view is particularly useful for clusters that capture an event or a discussion, as they can provide the evolution of the event as captured by its most dominant threads.

We discuss 6 of the clusters in Table 3.3 in more detail to show-case the kind of information that we can gain.

a. Detecting emerging security threats. First, several clusters consist of events that discuss novel security threats. For example, cluster 7 and 12 of OC revealed the growing concern of an extensive outbreak of the SimpleLocker ransomware and the RAT virus respectively. Also, cluster 2 of EH provides a timely warning of the explosive outbreak of Locky ransomware in Feb 2016.

b. Identifying bad actors and their tools. Our analysis can lead to important bad actors with Internet-wide reputation. Interestingly, it seems that hackers use their usernames consistently around Internet forums, possibly enjoying their notoriety. For example, our analysis (also shown in Table 3.3) leads to the usernames of hackers, "V4ND4l", "Dra-

gunman" and "VandaDGod". A simple Internet search of these usernames quickly leads to people with significant hacking activities and hacking tutorials on YouTube offered by them. Furthermore, we find that "VandaDGod" is active in multiple clusters in EH forum. In July 2019, a hacker group "VandaTheGod" is reportedly accused of defacing dozens of government sites [33].

3.4.2 Evaluation of TenFor

Evaluating the effectiveness of our approach and tool is inherently difficult due to the open-ended and subjective nature of the problem. We list our efforts to assess the precision and recall of our approach by examining the precision and recall to the best of our capabilities.

A. Precision. We present the evidence that our clusters are meaningful using several different angles. We find that **83%** of our clusters revolve around interesting events and each cluster shows high intra-cluster thread similarity. This is validated by a group of security experts and further corroborated via crowdsourcing and the REST methodology [52].

1. Manual evaluation from domain experts. We use a group of 3 security researchers to manually investigate all 52 clusters from all three forums. We asked the experts to (a) assign a score (out of 100) for each cluster based on the topic cohesiveness, and (b) summarize the important event(s) in each cluster, if they think the topic cohesiveness score crosses 70. Our experts determined 43 clusters containing 55 significant events based on the majority vote.

2. Manual evaluation via crowd-sourcing. We recruited nine judges among graduate students across campus to check the 52 clusters whether they contain noteworthy

events and assign a similar score per cluster like the domain experts. A key difference is that our volunteers make their decisions based on 10 randomly selected *thread titles* from each cluster. For calibrating their sensitivity, the judges were given two sample clusters before the evaluation with (a) randomly selected thread titles, and (b) titles from the same topic. Note that we declare a cluster as cohesive if at least five of the judges assign a topic cohesiveness score $\geq$70. The group declared 41/52 clusters (79%) as cohesive containing 56 events. For 52 clusters and 9 judges, we calculate the Fleiss' kappa score [49], $\kappa = 0.699$, which is substantial enough to come to a significant inter-annotator agreement in our context. In Table 3.2, we provide an overview of the results above. We argue that each combination in Table 3.2 columns has its own merit with the intersection being the most strict and the union being the more inclusive.

3. Assessing the cohesiveness. We corroborate the effectiveness of our content-based labeling (as A, T, P, G type) and assess the cohesiveness of clusters in an indirect way using a state-of-the-art technique, REST [52]. REST follows a thread-centric approach and labels threads along these four categories focusing on the content of a thread. We applied REST for every thread in our clusters. We find that 42 clusters have more than 70% threads of the same type according to REST and they also agree with our cluster label. Note that REST operates at the level of a thread, while we label clusters, which will inevitably introduce "errors". Thus, we consider the above matching numbers as a good indication for both: (a) the cohesiveness of our clusters, and (b) the accuracy of our labeling approach. Going one step further, we manually investigated the threads that REST was not confident enough to label. We randomly selected 200 such threads and found that 81%

Expert	Crowd	Expert AND Crowd	Expert OR Crowd	REST
83%	79%	71%	88%	79%

Table 3.2: Precision of TenFor: Percentage of clusters declared as interesting and cohesive in our evaluation.

Forum-CID	No. Users	Type	Top Threads	Top Users	Top Dates	Events and Explanation
OC-7	34	P	3502, 4843, 4841	S. Prajapati 23, Cyberseason, Assassin	Dec 2 & 15 2015, Feb 13 2016	(a) A market of 34 buyer/sellers of decryption tools against SimpleLocker ransomware with peaks in Dec 2015 and again in Feb 2016, which mirrors the outbreak events of SimpleLocker.
OC-8	54	A	2562, 1228, 1234	V4nD4l, RF, Pratham	Feb 4, 19, & 28 2016	A peak is detected when V4nD4L claimed success in hacking Facebook in Feb, 2016.
HTS-6	39	T	1125, 234, 6788	Ninjex, Rajor, mShred	Apr 7, Aug 22 & 31 2014	A peak in activities is observed when Ninjex and mShred shared tutorials for building hacking tools.
HTS-12	18	P	3453, 4467, 8901	whacker, DoSman, Bhaal	April 10 & 28, May 12 2016	DoSman offered a 30 days free trial of a DoS attack tool with a peak in April, 2016.
EH-2	31	A	7263, 8762, 9127	DarKnight, Don, VandaDGod	Feb 1 & 9 2016, May 15 2017	DarKnight was a victim of Locky ransomware in Feb 2016. WannaCry ransomware created a huge fuss in May 2017.
EH-6	46	T	1251, 8325, 8338	D3vil, VandaDGod	Nov 19 & 24 2017	VandaDGod, a expert Linux hacker, shared a popular tutorial series of hacking in Kali Linux in Nov 2017.

Table 3.3: Investigating nine clusters identified by TenFor reveals interesting activities. (CID is the id of the cluster).

of these threads were aligned with the type of the clusters they were in. Many of these threads were short, and we suspect that REST did not have enough context to assign a label.

B. Recall. Quantifying the recall of our approach is even harder. As answering to *"Are we missing important activities and events?"* question is harder to prove, we attempt to argue in favor of our method by providing three types of observations.

1. Any spike in activity is caught by TenFor. We argue that any event that creates significant activity involving threads and users will be caught by TenFor. To provide evidence, we find the top 20 weeks of high activity (in # posts), and the top 50 active users and threads (in # posts) in forum OC. We find that 19 out of the 20 most active weeks, 47 out of the 50 most active users, and 46 out of the 50 most active threads are also identified among the top 5 "performers" in our clusters (k=5).

2. Several real-life events are caught by TenFor. TenFor manages to capture several significant data breach events in the clusters from HTS forum including a) Sony Pictures, (b) Snapchat, and (c) Slack data breach. Users of security forums tend to be more interested in malware and ransomware discussions. For example, among the six most widespread ransomware from 2013-2017 listed in [20], TenFor captured 5 of them in 4 clusters: (a) SimpleLocker(2015-16) event in OC, (b) Locky(2016) and WannaCry(2017) ransomware event in EH, and (c) CryptoLocker(2014) and Petya(2016) ransomware in OC. Therefore, we argue that significant real-life events which discussed in the forums extensively are captured in the clusters.

C. Comparison with state-of-the-art methods. We compare TenFor with TimeCrunch [155], which identifies temporal patterns in a dynamic graph. This is the closest state-of-the-art method: our input tensor can be seen as a dynamic bipartite graph. We argue that TenFor is able to find more and meaningful cluster patterns compared to TimeCrunch.

Specifically, applying the default parameter-free setting of TimeCrunch, we find a total of 17 temporal patterns from three security forums, whereas TenFor finds a total of

52 clusters patterns. First, upon further investigation, we find that 13 of these 17 temporal patterns are actually present in our identified clusters. TimeCrunch reports only fixed types of patterns (full/near bipartite core, full/near clique, ranged/constant star etc.) based on Minimum Description Length (MDL) after encoding the model and the output patterns. Encoding larger clusters leads to higher MDL cost, which may be why TimeCrunch reports clusters of smaller sizes. TenFor does not consider any fixed types of pattern types and leverages the power of tensor decomposition. Furthermore, we observe that all 17 clusters are small in size (less than 21 users), compared to the TenFor cluster sizes (as much as 228 users). It seems that TimeCrunch does not identify larger clusters- probably can not "summarize" efficiently and, therefore, does not identify the interesting larger clusters.

We also compare TenFor with a widely-used community finding algorithm for Weighted Bipartite Network (CFWBN) [6]. This approach operates on the user-thread space and identifies a total of 771 bipartite communities from all three forums. However, we find 91% of these clusters are small, with ≤ 3 users, and only 35 communities start becoming substantial with ≥ 5 users. We argue that this large number of communities and the absence of time dimension make a follow-up investigation harder for the end-users.

In conclusion, TenFor strikes a balance between reporting too many and too few meaningful clusters compared to previous other methods. Additionally, it provides the end-users with key actors and a timeline of key events in an informative visualization.

D. Generalizability. We wanted to see if our approach would work equally well on different types of online forums of larger size. For this reason, we apply our method on our online gaming forum, MPGH, discussed in Section 3.2. Applying TenFor on this

forum, we find 41 clusters with a total of 1.3K users and 3K threads. Apart from finding clusters related to gaming strategy, and tricks for different popular online games, we also find several cyber-crime related activities even in this gaming forum! We highlight the indicative findings below.

(a) Scamming and cheating. Interestingly, the biggest cluster with 300 users and 400 threads is focused solely on scamming. The key perpetrators are reported to be Nigerian scammers and a well-known scamming company, "iYogi".

(b) Romance scamming. We identify a sudden emergence of "romance scamming" reports in the mid of August 2018. Apparently, scammers engage in online games, connect with other players, and win their affection and trust, which they use for monetary gain [24].

(c) Hacking for hire. Another surprising behavior is the search for a hacker to exact revenge on a gaming rival, as captured in a cluster with 69 users and 119 threads.

E. Computational effort. The computation required by TenFor is not excessive. The average runtime for preparing the final StoryLine View of the biggest forum with 100K posts, MPGH, takes only 4.35 minutes on average. Our experiments were conducted on a machine with 2.3GHz Intel Core i5 processor and 16GB RAM. We use Python v3.6.3 packages to implement all the modules of TenFor. We believe that the runtime can be reduced to seconds if we use more powerful hardware. These results suggest that TenFor scales reasonably well in practice.

3.5 Related Work

Overall, none of the previous efforts combines: (a) using tensor decomposition, and (b) extracting events of interest in an unsupervised manner. The most related work to the best of our knowledge is TimeCrunch [155]. TimeCrunch leverages the MDL principle and is limited to reporting only six fixed types of temporal patterns. It also does not use tensor decomposition and does not include a systematic event extraction mechanism like we do here. We discuss other related works briefly below.

a. Mining security forums: Some recent studies focus on identifying key actors and emerging concerns in security forums using supervised techniques and NLP by utilizing their social and linguistics behavior [112]. Some of these works are empirical studies without developing a systematic methodology. Recent efforts include analyzing the dynamics of the black-market of hacking services [137], extracting malicious IP addresses reported by users in security forums [51]. A recent work [52], REST, identifies and classifies threads given keywords of interest, and we use it to validate our cluster labeling.

b. Mining social networks and other types of forums: Researchers have studied a wide range of online media such as blogs, commenting platforms, Reddit, Facebook etc. Some recent works analyze the user behavioral patterns observed in Reddit [166] and infer information for the users from their activities on Facebook [12] and GitHub [145, 146, 72, 73]. Despite some common algorithmic foundations, we argue that different media and different questions require novel and targeted methods. Event detection is a broad and related type of research [160, 63, 84]. A recent work [114] proposes a hierarchical multi-

aspect attention approach for event detection but does not consider the author and temporal dimension as we do here.

c. Tensor Decomposition approaches: Tensors is a well-studied area with a wide range of diverse applications and domains [91] including understanding the multilingual social networks in online immigrant communities [130], community assignment of nodes in multi-aspect graph [61], and tensor-based community evolution [107]. We are not aware of any tensor-based event extraction studies for online forums. In our work, we adapted the CP tensor decomposition [91, 47] and combined it with L1 regularization to filter out the insignificant entities.

3.6 Conclusion

We propose, TenFor, an unsupervised-learning tensor-based approach to systematically identify important events in a three-dimensional space: (a) users, (b) threads, and (c) time. Our approach has three main advantages: (a) it operates in an unsupervised way, though the user has ways to influence its focus, if so desired, (b) it provides visual and intuitive information, and (c) it identifies both the events of interest, and the entities of interest within the event, including threads, users, and time intervals.

Our work is a step towards an automated unsupervised capability, which can allow security analysts and researchers to shift through the wealth of information that exists in security forum and online forums in general.

3.7 Acknowledgement

This work was supported by the UC Multicampus-National Lab Collaborative Research and Training (UCNLCRT) award #LFR18548554.

Chapter 4

RecTen: A Recursive Hierarchical Low Rank Tensor Factorization Method to Discover Hierarchical Patterns from Multi-modal Data

4.1 Introduction

Tensor decomposition has emerged as a powerful analytical tool with a rich variety of applications, but it focuses on identifying latent clusters without exploring any hierarchical structure that may exist. Tensors generalize the concept of a 2-dimensional matrix into multiple dimensions and we use the term *modes* to refer to these dimensions. On the one hand, current tensor-based approaches do an excellent job of identifying latent patterns in

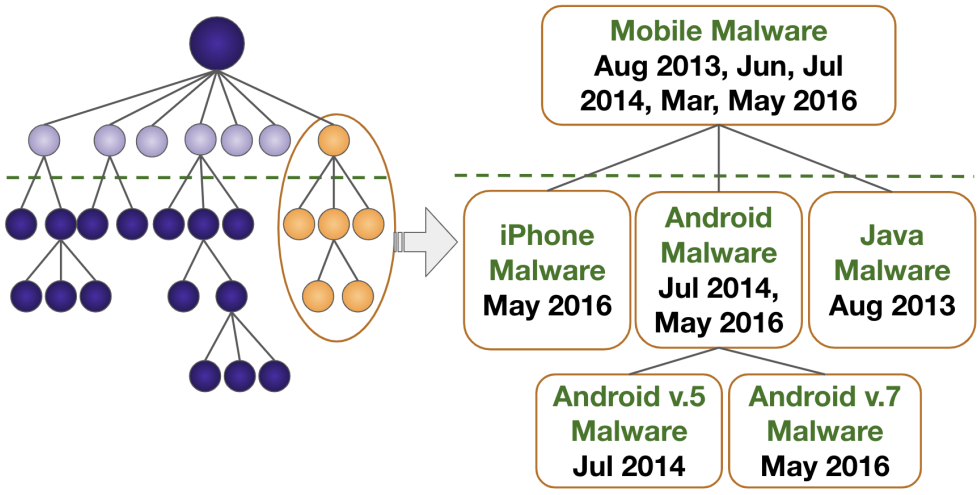


Figure 4.1: Output from RecTen after applying it on "Hack This Site" security forum data. The inset shows the structure of users and threads that focus on sub-topics of mobile malware, which is the focus of the parent cluster. Our decomposition also discovers the dates when the clusters are most active.

the form of soft clusters (an entity can belong to multiple clusters) and have been used successfully in a wide range of types of data across many disciplines. On the other hand, we argue that often behaviors and phenomena have an inherent hierarchical structure, which can provide interesting insights once it is revealed.

Problem: The main focus of this work is a relatively unexplored question: how can we extend tensor decomposition to identify hierarchies when such hierarchies exist in the data? We are given a multi-dimensional dataset, and we seek to identify clusters and their potentially hierarchical structure present in the data. The challenge here is twofold: (a) we do not know a priori anything about the data, such as the number of clusters or levels, (b) we want to adapt to different levels of "sensitivity" meaning that different parts of the data may hide more layers of hierarchy than others. We want to build on the power of tensor decomposition by expanding it to address the above challenges.

Formally, we state the problem as follows:

> Given a Tensor T, how can we expand the decomposition of T in a recursive manner into a hierarchy of clusters, C_i? The input is a tensor and the output is a hierarchy of clusters.

Algorithmically, the problem poses two main challenges: (a) we want to recursively decompose clusters in each layer of the hierarchy, and (b) we want to identify the right conditions for terminating this process.

Motivating case-study: *What applications would benefit from such a capability?* Apart from being an interesting theoretical problem, we can think of several applications that could make use of an effective solution. Here we showcase a focused example. A security analyst wants to identify the origin, evolution, and developer interactions for a particular malware. The input data is a large number of online security forums where, surprisingly, emboldened hackers openly sell malicious tools and services, while they boast of and collaborate on malicious acts [52, 74]. Each forum can be seen as a three-dimensional tensor with three modes: (a) users, (b) threads, and (c) time. The analyst initially wants a quick birds-eye view of the group activity of the users on each platform. Subsequently, she can focus on the threads that contain keywords/discussion of interest and conduct a more focused analysis, for example, evolution of these group activity in an iterative fashion. Fig. 4.1 provides a real example of the first step described above for the security forum "Hack This Site", which reveals the interesting and rich underlying structures with 27 clusters and 4 levels of hierarchy. A non-hierarchical approach would report only one level of detail e.g.

only the first line of clusters above the green dashed line in the figure. In the *Discussion* section, we elaborate further on the motivation and real-world applications of multi-modal hierarchical clustering.

Despite the vast literature on tensor decomposition, we are not aware of any work that fully explores the hierarchical tensor decomposition. We can group prior efforts into three main families: (a) hierarchical clustering in 2D matrices , (b) deep learning-based hierarchical clustering , and (c) non-hierarchical tensor decomposition clustering. We discuss the prior works in detail in the *Related Work* section, while we provide a qualitative comparison of these approaches in Table 4.1.

Contribution: As our key contribution, we propose RecTen, a hierarchical soft clustering approach based on tensor decomposition. Our approach provides the required mechanisms for recursively decomposing clusters, and for terminating this recursive process. We evaluate our proposed algorithm using both synthetic and real data. We use synthetic data to evaluate the performance given the absence of an established benchmark. We also use this synthetic data to evaluate the sensitivity of three internal parameters, which enables us to provide recommendations for hands-free operation.

a. RecTen compares favorably against the state-of-the-art algorithms. We find that RecTen performs favorably when compared to six other state-of-the-art methods, as shown in Table 4.2, using our synthetic hierarchical data. Interestingly, RecTen performs well even with flat (non-hierarchical) data as shown in Table 4.3.

b. RecTen extracts meaningful clusters with real-world data. We apply RecTen on four online forums and a dataset that represents user interactions on GitHub.

This provides indirect evidence of the usefulness of RecTen in identifying meaningful clusters, such as tight-knit groups of users, and events. For example, the discovered clusters (27 clusters, across 4 levels of hierarchy) of "Hack This Site" forum in Fig. 4.1 are meaningful in the sense that they identify communities of special interest and point out the activity peaks in time.

A usable open-source platform for maximal impact. As a tangible contribution, we implement RecTen as a powerful user-friendly platform that will be useful to researchers, and practitioners. The platform expects a Tensor as input and produces an output hierarchy of clusters which can be analyzed further to understand the hidden structures. The key advantages of RecTen platform is that it is user-friendly by being both automatic and customizable. RecTen can run with default parameter settings, but savvy end-users can optionally tune the parameters based on their needs and preferences.

We will make both our platform and datasets available that can help establish research benchmarks.

4.2 Background and Datasets

We provide some fundamental concepts, a qualitative evaluation, and a description of our datasets.

4.2.1 Background

We discuss the tensor decomposition and the algorithmic landscape below.

Tensors and decomposition. Tensor decomposition has been established as a powerful analytical tool. In recent years, it has been the basis for many algorithmic solutions and many practical applications [91, 107, 130, 76]. We have discussed the basics of tensor decomposition in Section 3.3.1 of Chapter 3.

The algorithmic landscape. We provide a high-level overview of the algorithmic landscape with respect to multi-dimensional hierarchical clustering. We can consider the following families of approaches: (a) 2D matrix methods, (b) Deep Learning-based methods, and (c) tensor-based methods but without hierarchy support. A qualitative analysis is provided in Table 4.1. In a nutshell, 2D matrix approaches are limited in dimensions, and in our context often miss the extra temporal dimension [11, 52]. Deep Learning approaches require large datasets to work well and often the results are less intuitive to interpret and explain [87]. Finally, to the best of our knowledge, tensor-based approaches so far have not supported hierarchies. A more detailed discussion of previous works is provided in the *Related Work* section.

4.2.2 Datasets

In our evaluation, we consider the following datasets: (a) security forums, (b) gaming forum, and (c) a group of GitHub repositories of malware software and their authors mentioned in Chapter 2.

Family	Sample related works	Multi-modal ($\geq$ 3D)	Performance on small data	Interpre -tability	Detecting hierarchy
2D matrix	[179]	✗	✓	✓	✓
Deep Learning	[87]	✓	✗	✗	✓
Tensor Decomposition (Non-hierarchical)	[74]	✓	✓	✓	✗
RecTen	This work	✓	✓	✓	✓

Table 4.1: Overview of the related algorithmic landscape: a qualitative assessment.

4.3 Our Approach

We present, RecTen, a novel tensor-based multi-step recursive approach that identifies patterns in an unsupervised way. Fig. 4.2 provides the high-level pseudo-code of the basic workflow. Fig. 4.1 provides the sample output of RecTen. Conceptually, our approach works in three steps. First, we decompose a tensor into clusters at level 1. Second, we "perturb" each cluster at the current level which we consider as another tensor to be decomposed further. Third, we have two termination criteria that stop this recursive process. We explain each step below.

4.3.1 Step 1: Tensor-based Clustering

As a first step, we apply CP decomposition on the given input tensor. We provide a quick overview of the challenges and algorithmic choices in the decomposition algorithm below.

a. What is the ideal number of components to target in the decomposition? To answer this question, we use the AutoTen method [131] and find the rank (R) of the tensor, which points to the ideal number of clusters to be decomposed into. AutoTen attempts to

identify the solution that extracts a large-enough number of components while maintaining a high core consistency, which is a metric for model appropriateness/goodness.

b. How can we decompose the tensor? We use the non-negative Canonical Polyadic, also known as CANDECOMP/ PARAFAC (CP), decomposition to find the clusters. RecTen achieves this non-negative factorization by adding the non-negative constraint in CP decomposition.

c. How can we strike a balance on cluster size? Each cluster, derived from a 3D tensor, is defined by three vectors whose lengths are equal to a dimension of the tensor as shown in Fig. 3.3 of Chapter 3. However, RecTen provides the functionality of having clusters with significant elements only. Therefore, we need a threshold to determine significant participation in the cluster, which is a common practice for (a) avoiding unreasonably dense clusters [150], (b) enhancing interpretability, and (c) suppressing noise. So, the challenge is to impose this sparsity constraint and eliminate the need for ad-hoc thresholding to find the clusters with only significant entities. Our solution is to add L_1 norm regularization with non-negative CP decomposition. L_1 regularization pushes the small non-zero values towards zero. Therefore, for each vector, we filter out the zero-valued elements and produce clusters with significant elements only. In this way, we eliminate the noisy entities having the least significant contributions in the cluster. The final model that we use for finding the clusters looks like this:

$\min_{A\geq 0, B\geq 0, C\geq 0} \|X - D\|_F^2 + \lambda(\sum_{i,r}|A(i,r)| + \sum_{j,r}|B(j,r)| + \sum_{k,r}|C(k,r)|)$ where λ is the **Sparsity Regularizer Penalty** and $D = \sum_r A(:,r) \circ B(:,r) \circ C(:,r)$. To find the clusters, we solve the above equation. Since the equation is highly non-convex in nature, we use the well-established

Alternating Least Squares (ALS) optimizer as the solver. An example of a cluster after filtering is shown in Fig. 3.4 of Chapter 3.

4.3.2 Step 2: Processing for Next Level Decomposition

Having obtained the clusters from the previous level, our goal is to further decompose each cluster, if termination conditions are not met. Intuitively, the idea is to introduce some perturbation in the cluster to help reveal a structure that is currently avoiding detection.

Rank modification. The main mathematical challenge in this phase is to answer the question: "*How can we get the clusters ready to be decomposed further for the next level?*" Recall that the rank of every cluster is 1, which is why these clusters have not already been decomposed. To decompose a cluster at level l, we need the rank of the cluster to be greater than one. We propose to achieve this by introducing a small perturbation in the cluster by zeroing-out some tensor elements, which changes the rank of the cluster to ≥ 1.

We can illustrate the intuition behind this process with the following simple example . The top-level decomposition provides a set of a rank-one clusters. Let us assume that one such cluster contains two smaller "sub-clusters" within it, but there are enough interactions between these sub-clusters so that the top-level decomposition assumes that this is best represented as a single cluster of rank one. By appropriately removing a few connections between the sub-clusters, we can reveal the underlying structure, which will "push" the rank of the cluster to 2. Namely, the two sub-clusters have become sufficiently distinct for the next-level decomposition. We discuss more about this in *Discussion* section along with experimental results.

Choosing the "target elements" to zero-out. The natural next question is: "*Which elements to choose for zeroing out?*". We propose to select the non-zero valued elements stochastically, but with a bias towards elements with low numerical value.

Let's assume C_l to be a rank-one cluster at level l, $l \geq 1$. We get the processed cluster $\hat{C}_l$ with rank, $r >= 1$, in the following manner. We choose a subset of non-zero elements, $C_l(i, j, k)$, stochastically favoring elements with low numerical value. Specifically, the probability of selecting an element is inversely proportional to its numerical value $C_l(i, j, k)$. That means, the higher the element value, $C_l(i, j, k)$, the lower the probability of getting replaced with 0. Formally, for a 3-mode tensor, the probability of selecting an element, e, having value w among the non-zero elements of cluster C_l is given by the formula:

$$P(\text{e is chosen}) = \frac{\frac{1}{w}}{\sum_{\forall z \in C_l} \frac{1}{z}}$$

where z represents each non-zero value in C_l.

Determining the Deletion Percentage, ϵ : The next question that arises is the following: *"How many elements should we zero-out?"*. We introduce the **Deletion Percentage** parameter, ϵ, which determines the percentage of the total non-zero valued elements in the cluster which we zero-out for a cluster. More precisely, we get the ceiling of that number to ensure that it is an integer, but we also never zero-out all non-zero elements. In our *Evaluation* section, we study the sensitivity of our approach to the Deletion percentage parameter. In addition, we discuss whether these perturbations introduce artifacts in our *Discussion* section.

Algorithm 1:

RecTen (T) Algorithm to factorize a Tensor recursively to have hierarchical clusters.

Input: Root Tensor, T
Output: Clusters arranged in hierarchical tree format

```
Clusters = [clusters from first level Decomposition of
  given root tensor, T]
Final_clusters_tree.insert(T,null)
Final_clusters_tree.insert(Clusters,T)
while Clusters != empty do
    Temp_clusters=[]
    for each C in Clusters do
        if Termination Condition 1 then
            continue
        end
        Construct Ĉ by removing some elements
          from C
        if Termination Condition 2 then
            continue
        end
        Decompose Ĉ into new clusters C_i's
        Temp_clusters.insert(C_i's)
        Final_clusters_tree.insert(C_i's, C)
    end
    Clusters=Temp_clusters
end
return Final_clusters_tree
```

Figure 4.2: Algorithm to factorize a Tensor recursively to have hierarchical clusters.

After assigning zero values to the selected elements, we obtain a perturbed cluster, which we will consider for decomposition in the next level as long as it does not meet the termination criteria, which we discuss below.

4.3.3 Step 3: Termination Condition

We stop the recursive procedure of clustering, when one of the two termination conditions is met:

a. Termination condition 1: cluster size. This termination condition suggests that when the cluster/tensor size is relatively small, we do not attempt to further decompose it. Now the obvious question is: "*When do we call a cluster relatively small?*" Note that we use the concept of cluster size to refer to the number of non-zero elements of the cluster. We introduce the **Minimum Cluster Size** parameter, **k**, which determines that: we do not decompose a cluster further if its size is less than k percent of the average size of its sibling clusters at the same level.

Intuitively, this criterion gives us the ability to provide a flexible mechanism to contain the depth of the recursive decomposition. Naturally, there are many different ways to specify such a condition, including a hard size limit. Here, we opted to specify it as a percentage of the sizes of the clusters of the level to allow for some "self-adaptation". We study the effect of this parameter in our *Evaluation* .

b. Termination condition 2: rank=1. Naturally, the recursive factorization cannot continue if the rank of a cluster is one, even after the perturbation. Therefore, the second terminating condition is: after perturbation, if AutoTen returns rank=1, we stop.

Claim 1. In RecTen, zeroing-out a strict subset of non-zero elements from the rank-1 tensor changes the rank to ≥ 1.

We present the intuition behind this claim here. Assume that zeroing-out some non-zero elements of tensor T leads to a new tensor $\dot{T}$ of rank zero. We want to prove $rank(\dot{T}) - rank(T) \geq 0$. By definition, a rank-zero tensor has only zero elements. Thus, $\dot{T}$ should have only zero elements. However, this introduces contradiction. It is not possible

to have a zero-tensor as we only zero-out a strict subset of the non-zero elements. Thus, zeroing out can never lead to rank-0 tensor, i.e. $rank(\dot{T}) - rank(T) \not< 0$). Note that formally, we can represent the zeroing-out process using element-wise Hadamard Product between the rank-one tensor, T, and a basis tensor B i.e. $\dot{T} = T. * B$ where B is the basis tensor with elements either set (1) or reset (0).

The description of our approach is likely to generate the following questions to an astute reader:

a. Is the perturbation introducing an artificial hierarchical structure? We answer this question in the *Discussion* section.

b. How sensitive is the performance of the approach to its three main parameters? We answer this question in the *Evaluation* section, where we study the effect of the three parameters: (i) Deletion Percentage, ϵ, (ii) Minimum Cluster Size, k, and (iii) Sparsity Regularizer Penalty, λ.

4.4 Evaluation

As RecTen is a multi-modal hierarchical soft clustering method, a thorough evaluation is challenging due to: (a) there is a lack of established ground truth, (b) there is not a well-established methodology for generating realistic synthetic data, (c) finding suitable evaluation metrics is non-trivial, and (d) it is not obvious what are the most appropriate reference methods. We present our efforts to address these challenges below.

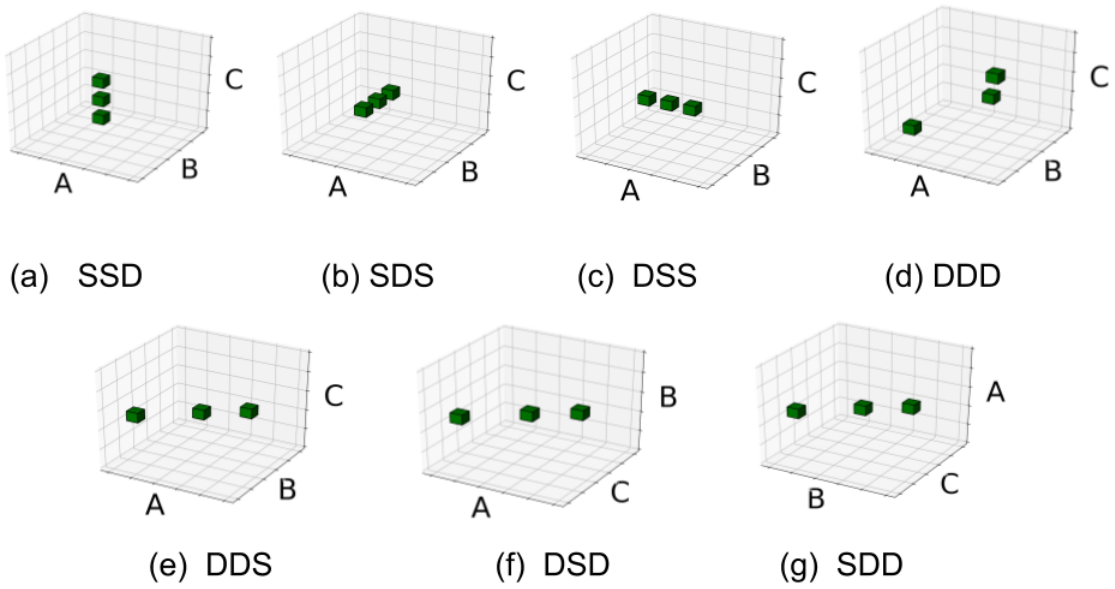


Figure 4.3: D_Flat: Creation of challenging (overlapping in 2-modes) clusters in our synthetic tensor by combining the depicted 21 clusters.

4.4.1 Synthetic Tensor Construction

To evaluate RecTen, we use flat and hierarchical synthetic 3-mode tensors, which we describe below.

A. D_Flat: a non-hierarchical synthetic tensor. We generate a flat (non-hierarchical) 3-mode tensor for evaluation purpose. The advantage of a synthetic tensor is that they have a well-established ground truth. To stress-test our algorithms, described later, we generate a 3-mode synthetic tensor, D_Flat. The dimension of D_Flat is $300 \times 300 \times 30$, which we find sufficient for our evaluation.

To elaborate, we start from a zero-tensor, Z. Let us consider that Z has three modes A, B, and C, with indices a_i, b_j, and c_k along the modes respectively. Then we insert some clusters in Z in such a way that these clusters are not decomposed into further

clusters. We call these clusters flat because they span in level 1 only.

Fig. 4.3 shows that a total of 21 clusters (3 clusters from each of the 7 groups) have been introduced which forms the ground truth. Some of the clusters "overlap", if they get projected in only two dimensions.

The three-letter notation, e.g. SSD, indicates the mode along which the clusters have similar (S) or different (D) values in the corresponding dimension. For example, the three inserted SSD clusters in Fig. 4.3(a) have the same a_is and b_js, but different c_ks meaning that the three clusters contain same members (across A and B modes) but evolve in different times (along C mode).

How do we insert (i.e. add elements to) each cluster? We identify a center for each cluster and then arrange nodes (equivalently, non-zero elements) around that center by finding the position to insert stochastically. We introduce four parameters to control the size, and other properties of these clusters, which we refer to as **Synthetic Cluster Construction Parameters**. The number of nodes per cluster is controlled by the **concentration** parameter ρ while the **cluster radius**, d, determines the radius of the cluster. The value for each element is drawn from a Gaussian distribution, $G(\mu = 10, \sigma = 3)$.

B. D_Hi: a hierarchical synthetic tensor. The goal here is to generate a 3D hierarchical tensor. To do this, we use a two-dimensional Kronecker graphs [102], which have well-defined and controlled hierarchical properties and introduce a third dimension that emulates a temporal evolution partially inspired by previous work [60].

Specifically, we create a 2D hierarchical Kronecker matrix, and then we create a series of **slices**: each slice is a slightly modified version of the previous slice. The con-

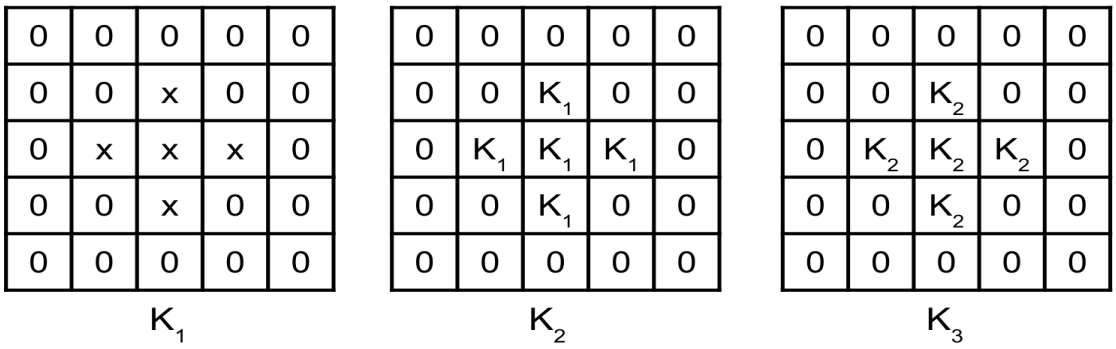


Figure 4.4: Example of Generation of Kronecker adjacency matrix K_3 for the base slice where $K_3 = K_2 \otimes K_1$ and $K_2 = K_1 \otimes K_1$. $\otimes$ is the Kronecker Multiplication operator.

centration (stacking) of these slices creates the third dimension that imitates an evolving network.

To elaborate, we first create a hierarchical graph utilizing Kronecker Multiplication of order three (K_3 adjacency matrix) in the base slice (dimension 125X125) demonstrated in Fig. 4.4. The values of non-zero elements (denoted as x) in this figure are drawn from a Gaussian distribution, $G(\mu = 3, \sigma = 1)$, while all the other elements have a zero value. Thus, the base slice has a two-level hierarchy with 5 clusters at level 1, and each cluster consists of 5 sub-clusters. Second, we create slices, which we limit to 10. For each new slice, we randomly choose $n\%$ of the total data points of the previous slice and assign them new values drawn from the aforementioned Gaussian distribution $G(\mu = 10, \sigma = 3)$. In this way, we create a 125x125x10 tensor with an underlying two-level hierarchical structure.

We discuss the effect of the noise on the performance of RecTen below in this section.

4.4.2 Evaluation Metrics

Evaluating hierarchical multi-modal clustering is challenging as its quality can be analyzed from several different perspectives. For consistency, we adopt metrics from previous methods [108, 142, 194] which we present below.

A. Total Purity. Total Purity (*TP*) [108] captures the quality of the clustering and it is measured on a scale of 0 to 1 where *TP=1* indicates perfect clustering. Intuitively, TP represents the percentage of nodes that are associated with the correct cluster and assumes the existence of ground-truth.

B. Rand Index. The Rand Index (RI) [142] is a measure of similarity between two clustering algorithms on the same data. The metric considers all pairs of elements and counts pairs that are assigned in the same or different clusters by each algorithm. RI has a value within [0,1]. A value of 1 represents identical clustering solutions.

Given a set, S, of n elements and two clustering algorithms, X and Y, to compare, the formula to calculate RI is:

$$RI = \frac{a+b}{\binom{n}{2}}$$

where a is the number of pairs of elements in S that are in the same cluster for X and in the same cluster for Y. b is the number of pairs of elements in S that are in the different clusters for X and in the different clusters for Y. In our case, n denotes the total number of non-zero elements in the synthetic tensor.

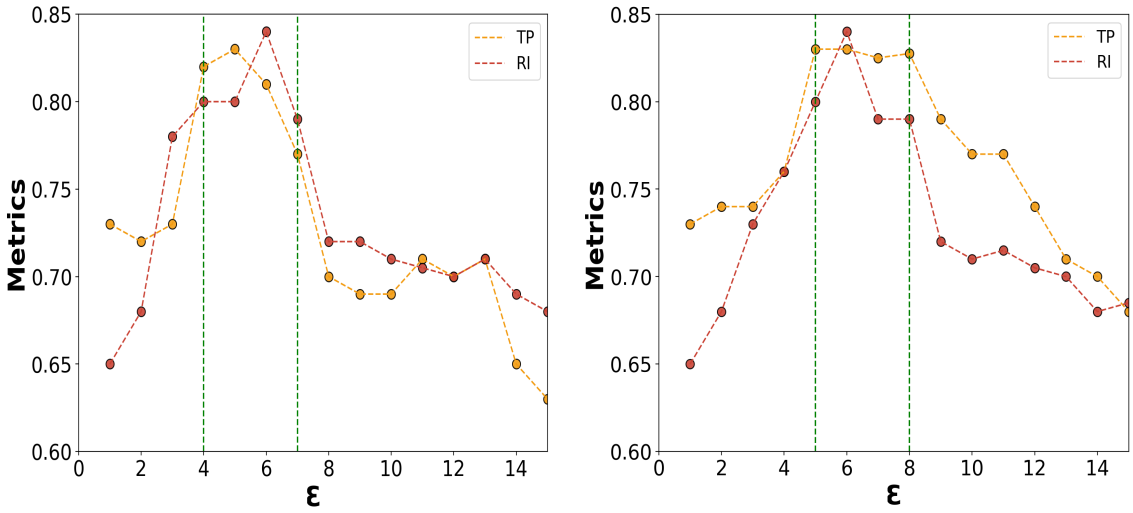
C. Tree Edit Distance. The Tree Edit Distance (TED) [194] measures the similarity of two trees. Here, we can represent a hierarchical clustering by a tree and use this metric. Given that member elements have identities of which cluster it belongs to in the ground truth, we label each node of the tree with the majority members' identity. We use the concept of labeled trees to distinguish between tree nodes.

The TED metric of two labeled trees, T_1 and T_2, is the number of insertion, renaming and deletion operations needed to transform one tree into an exact copy of the other tree. The lower value of the TED, the more similar are the trees. Thus, when compared with ground truth, low values of TED are preferable.

4.4.3 The Sensitivity to Algorithmic Parameters

We assess the sensitivity of the performance of our approach to the three algorithmic parameters using Total Purity, and Rand Index metrics.

a. The sensitivity of RecTen to Deletion Percentage, ϵ. Choosing the right ϵ is crucial for RecTen. Very small ϵ may yield less clusters whereas very large ϵ may end up in extracting too many clusters. In both cases, the extracted clusters may not be meaningful. So, the goal is to find a sweet-spot, where we can unravel meaningful patterns with the least amount of deletion. Fig. 4.5 suggests that $\epsilon \in [4\%, 8\%]$ is our sweet-spot, where we achieve maximum performance based on both metrics, TP and RI, for both non-hierarchical and hierarchical cluster extraction from synthetic tensors. This implies that with reasonable amount of deletion, RecTen is able to extract reliable next level clusters.



(a) Non-hierarchical data.

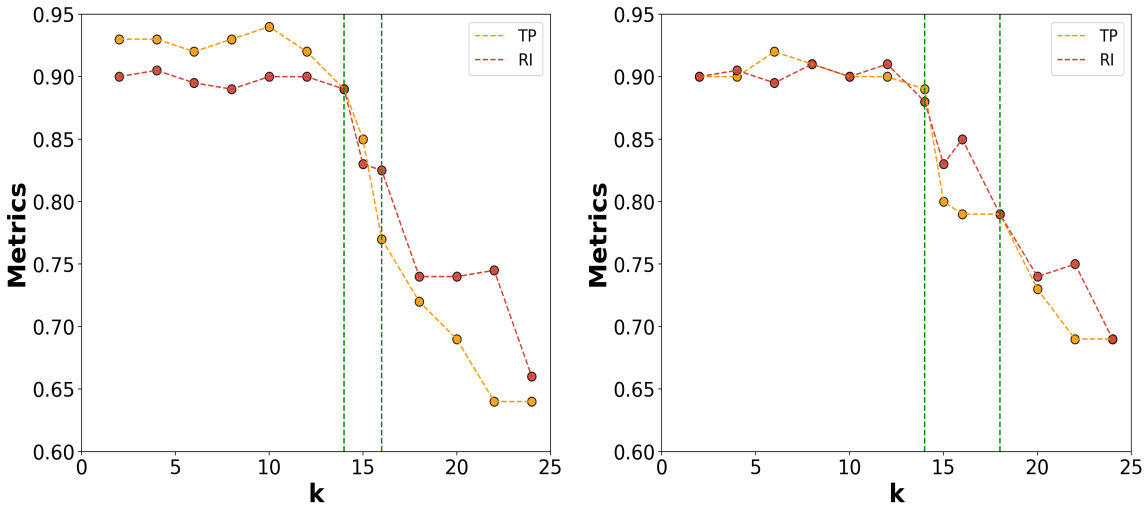
(b) Hierarchical date.

Figure 4.5: The effect of Deletion Percentage parameter ϵ on clustering quality metrics TP and RI ($k = 15$, $\lambda = 0, n = 10$).

b. The sensitivity of RecTen to Minimum Cluster Size, k. Another crucial parameter of RecTen is k which determines when to stop our recursive factorization. Very low k yields in small factorized clusters breaking down the pattern to even more parts (value of performance metrics close to 1) whereas very high value of k will preserve multiple convoluted patterns in a single cluster (value of performance metrics far away from 1). Fig. 4.6 exhibits that, for our synthetic tensors, both hierarchical and non-hierarchical, $k \in [14\%, 18\%]$ is our sweet region where we achieve reasonably high performance from RecTen based on both TP and RI performance metrics.

c. The sensitivity of RecTen to Sparsity Regularizer Penalty, λ.

The Sparsity Regularizer Penalty parameter, λ, is used to select cluster members during the decomposition as we explained earlier. Low values of λ create larger clusters, while high values create smaller clusters. Clearly, there is a need for a balanced solution



(a) Non-hierarchical data.

(b) Hierarchical data.

Figure 4.6: The effect of Minimum Cluster Size k parameter on clustering quality metrics TP and RI ($\epsilon = 6$, $\lambda = 0, n = 10$).

that will provide maximal information and insights from the data. Varying the value of the parameter in our study, we find that a value of λ to 0.8 provides the best results w.r.t. the Total Purity metric. The full results are omitted due to space limitations.

Practical guidelines for using RecTen. There are only three algorithmic parameters in RecTen: (a) Deletion Percentage ϵ, (b) Minimum Cluster Size k, and (c) Sparsity Regularizer Penalty λ. Based on the experience from our study, we recommend the following default values for these parameters: $k = 15\%$, $\epsilon = 6\%, \lambda = 0.8$. Even a savvy end-user can tune these parameter knobs to tailor them to the needs of their study. To recap, setting ϵ to a high value can enforce extracting higher number of clusters by increasing the rank of a tensor. A much higher value of k will terminate the decomposition for larger sizes, providing a lower bound on the size of the clusters. Furthermore, a high value of λ will affect the size of the clusters during the tensor decomposition. Overall, our results

suggest the following ranges for these parameters ϵ within 4%-8%, k within 14%-18%, and $\lambda = 0.8$.

4.4.4 The Sensitivity of RecTen to Dataset Properties

We wanted to evaluate the robustness of our approach to a wide range of data. An advantage of using the synthetic data is that we are able to create a wide range of datasets by varying the some parameters, e.g. Synthetic Cluster Construction Parameters. We briefly discuss their effects below. Unfortunately, the full set of results are omitted due to space limitations.

a. The effect of the amount of noise in the synthetic tensor construction: noise percentage, n. As mentioned earlier, we introduce some noise while creating each new slice while constructing our hierarchical dataset D_Hi. We vary the value of noise percentage, n, to stress-test the resilience of RecTen to this parameter. We find that RecTen can offer reasonable performance (TP=0.796, RI=0.8) for up to $n = 20\%$, at which point we observe a sharp drop in the performance.

b. The effect of the Synthetic Cluster Construction Parameters: d, ρ, μ, σ. We also analyze the performance of RecTen by varying radius d, concentration ρ, and data value distribution parameters μ and σ which affect the generation of our non-hierarchical synthetic tensor. We found that RecTen is relatively robust to different values of these parameters. For example, we found a case where doubling the parameter values did not change the performance significantly. Specifically, RecTen shows TP=0.85 for $d = 4$, $\rho = 8$/pattern, $\mu = 10$, and $\sigma = 3$, and drops to TP=0.835, when we double the parameter values to $d = 8$, $\rho = 16$/pattern, $\mu = 20$, and $\sigma = 6$.

Baselines	Slice 4		Slice 6		Slice 8		Slice 10	
	TP	TED	TP	TED	TP	TED	TP	TED
AHC_ward (2D)	0.79	6	0.75	8	0.7	8	0.65	10
AHC_freq (2D)	0.79	8	0.75	10	0.69	13	0.71	15
DLORE-DP (2D)	0.8	6	0.77	6	0.74	5	0.73	4
Affinity (2D)	0.77	6	0.73	7	0.72	3	0.7	3
TenFor (3D)	0.78	16	0.79	18	0.8	20	0.8	21
DynamicT (3D)	0.71	6	0.72	16	0.73	15	0.73	17
RecTen (3D)	**0.82**	**5**	**0.84**	**4**	**0.84**	**2**	**0.86**	**2**

Table 4.2: Performance evaluation of RecTen compared to baseline algorithms in terms of Total Purity (TP) and Tree Edit Distance (TED) metrics for hierarchical synthetic data D_Hi. We use bold for the best performance per column.

4.4.5 Comparison with State-of-the-art Methods

While there is not a widely-accepted set of baseline algorithms for multi-modal hierarchical clustering, we compare RecTen with a set of other widely-used and state-of-the-art which include both 2D hierarchical and 3D non-hierarchical methods. Specifically, we compare against the following approaches: (a) the widely-used basic Ward's method for Agglomerative Hierarchical Clustering Algorithm (AHC_ward) [179], (b) the frequency-based Agglomerative Hierarchical Clustering Algorithm (AHC_freq) [109], (c) the local cores-based Hierarchical Clustering Algorithm (DLORE-DP) [31], (d) the minimum spanning tree-based Affinity [11] clustering (e) the non-hierarchical TenFor [74], and (f) the non-hierarchical DynamicT [163].

Stress-testing: slices and noise. In the evaluation below, we will use the concept of the slice, which we can see as a snapshot in time of an evolving matrix. We introduced the idea of slice when we described the generation of our synthetic data earlier

Baselines	Slice 4		Slice 6		Slice 8		Slice 10	
	TP	RI	TP	RI	TP	RI	TP	RI
AHC_ward (2D)	0.79	0.81	0.75	0.77	0.74	0.76	0.7	0.71
AHC_freq (2D)	0.77	0.78	0.75	0.74	0.69	0.73	0.72	0.7
DLORE-DP (2D)	0.79	0.76	0.77	0.76	0.73	0.75	0.73	0.74
Affinity (2D)	0.77	0.76	0.72	0.7	0.71	0.73	0.71	0.73
TenFor (3D)	0.79	0.8	0.81	0.81	0.82	0.82	0.82	0.83
DynamicT (3D)	0.82	0.81	**0.82**	0.82	0.82	0.83	0.81	0.82
RecTen (3D)	**0.83**	**0.82**	**0.82**	**0.84**	**0.83**	**0.84**	**0.85**	**0.87**

Table 4.3: Performance evaluation of RecTen compared to reference algorithms. We have presented the results of Total Purity and Rand Index metrics for non-hierarchical synthetic data D_Flat.

in this section. Recall that, we start with a fairly pristine clustering structure in the base slice and continue stacking slices on top of the base slice. Therefore, a 3D tensor can be viewed as a stack of slices. In each slice, we add some noise, which can be seen abstractly as modifying the cluster members (e.g. reducing the membership strength of an element). The goal is to stress-test the capabilities of RecTen to extract the underlying clusters despite the new modifications.

RecTen outperforms competitions especially in the face of higher noise. Our comparison results suggest that RecTen outperforms other baseline algorithms. Interestingly, the difference becomes more pronounced as we increase the third-dimension, as we explain below. To achieve this, we leverage the concept of slices, as discsussed above. We apply the 2D algorithms on 2D slice i of the synthetic tensors, while the 3D algorithms are applied on the 3D tensor constructed from slices 1 to i. For instance, we produce a hierarchy from slice 5 (2D) using 2D AHC_ward, and produce another hierarchy using RecTen from the 3D sub-tensor constructed by stacking slices 1 to 5. For completeness, we compare with

both 2D and 3D algorithms as we explain below. The results from the 2D algorithms are offered *only as a reference point* and not as a head to head comparison. The results are reported in Table 4.2 for hierarchical data D_Hi and in Table 4.3 for the non-hierarchical data D_Flat.

The performance of RecTen improves as we consider more slices i in Table 4.2. For example, we see that the Total Purity increases from 0.82 to 0.86, and the Tree Edit Distance decreases from 5 to 2. This is not surprising: as the third dimension becomes longer, it increases the amount of information that the algorithm can use. We also see that RecTen outperforms the other 3D algorithms, TenFor and DynamicT, with respect to both metrics. For example, for slice 10 in Table 4.2, RecTen achieves a TP of 0.86, compared to 0.8 and 0.73 for the other algorithms. Similarly, RecTen has TED of 2 compared to 17 and 21. The performance difference of RecTen compared to other 3D methods is statistically significantly ($p > 0.05$).

RecTen performs well even with non-hierarchical data, which suggests that it does not "force" a hierarchy if such hierarchies do not exist in the data. The results are shown in Table 4.3. RecTen outperforms other methods in terms of both TP and RI (statistical significance $p > 0.05$). Interestingly here the difference between RecTen and TenFor and DynamicT is relatively smaller compared to the difference with hierarchical data.

4.5 Application Results and Observations

We provide proof of the effectiveness of RecTen by finding interesting hierarchical clusters from five different real datasets, discussed in *Background and Datasets* section. We utilized

the forum analysis tools and NLP techniques [74] to profile the clusters and verified that they are meaningful: cohesive and focused on a topic or event.

a. Results from Security forum datasets. Applying RecTen on three security forum datasets reveals interesting clusters.

First, we describe how we constructed the input tensor. We construct a 3D tensor, T, by capturing the interaction of users with different threads at different weekly discretized times. Each element, $T(i, i, k)$, of the input tensor captures the interaction (in terms of the number of posts) of user i with thread j at discretized week k or zero in the absence of such interaction. We then fed the input tensor, T, to RecTen. We find a total of 101 clusters from three security forums (41 from OC, 27 from HTS, and 33 from EH) arranged in hierarchical format. Second, we dig deeper into the identified cluster to find out the discussion topics of the clusters. Some of the key results are described below.

We find that RecTen indeed captures meaningful hierarchical clusters from OC, which we validate with prior profiling NLP tools and manual investigation. For instance, a cluster at level 1 revolving around Ransomware related discussion. The discussion started in Dec 2015 and was further instigated in Feb 2016 (observed from the time dimension of the identified cluster) which mirrors the outbreak of SimpleLocker ransomware at that time. The detected cluster actually indicated the early signs of the coming world-wide ransomware disaster. Upon further investigation of that cluster by going one level down, we identified a smaller cluster with 12 threads and 34 sellers of their decryption tools (to recover from the malware) in February 2016. By going into the next level in the hierarchy, we identify that a well-known company, MDS, was also selling the decryption tools.

Investigating one of the identified clusters at level 1 from the HTS forum, we identify a group of 32 threads by user *DoSman* offering a free trial of his attack tools. Diving deep into the next level, we find posts related to selling DOS attack tools and phishing tools in April-May 2016.

The EH forum also revealed interesting clusters. For example, we identify *VandaDGod*, an expert Linux hacker (or possibly a group of hackers), who shared a popular tutorial series of hacking in Kali Linux in November 2017. Going one step lower, we find clusters related to 'Hacking into Banks' and 'Hacking Routers'. We find a series of posts to recruit new members for hacking into banks by *VandaDGod* in the next level. A simple Internet search reveals that extensive and notorious reputation: *VandaTheGod* is accused of hacking several government sites.

We argue that the above results provide a strong indication that RecTen captures meaningful clusters and reveals interesting activities.

b. Results from Gaming forum dataset. We analyze the gaming forum MPGH, and we find a total of 233 clusters from MPGH organized in 6 levels. As expected, we find clusters related to gaming strategies for specific games, but we also found some unexpected clusters. For example, we identify a big cluster at level one revolving around different scamming and hacking-related objections. In the next level, this cluster consists of clusters revolving around online gaming account scamming and 'Romance Scamming'. It seems that in some online games where users can chat among themselves, scammers connect with other users to win their trust and extract money. Also, we identify clusters related to searching for experienced hackers after a painful defeat in a game.

These surprising findings indicate that online gaming forums are being a new potential source of security threats.

c. Results from GitHub dataset. Similar to security forums, we construct a 3D tensor for GitHub dataset. Each element, $T(i, j, k)$, of the input tensor captures the interaction (in terms of the total number of create, fork, comment and contribution performed) between: (a) author i, (b) repository j, (c) per week k. Applying RecTen on this tensor, we extract a total of 79 clusters in 4 levels. An interesting cluster contains Windows related malware in the first level and the next level contains Windows related ransomware. These clusters formed mainly in Jan 2016 when ransomware was spreading worldwide and malicious authors started developing more ransomware in GitHub inspired by the attack success. RecTen can help the security enforcement authorities to keep track of which and how malware are being developed and getting popularity over time.

Empirical results comparison with TimeCrunch. TimeCrunch [155] is also a tensor-based tool to discover patterns, but it extracts only six fixed types of temporal structures, such as near cliques, bipartite cores, and spikes. We apply TimeCrunch on three security forum datasets and find a total of only 17 structures. All these 17 structures are captured in RecTen as well. Moreover, RecTen captures a total of 101 clusters from these security forums. We think that this suggests RecTen strikes a good balance between finding too few and too many clusters of interest. We omit the detailed finding of TimeCrunch due to space limitations.

Computational effort. The computation required by RecTen is not excessive. The average runtime for preparing the final hierarchical Tree view of the biggest forum with

100K posts, MPGH, takes only 2.39 minutes on average whereas TimeCrunch takes 1.98 minutes on average. We reason that TimeCrunch is faster because it operates in single-level decomposition, whereas RecTen performs a recursive multi-level decomposition. In that sense, the computational effort of RecTen seems reasonable. However, we intend to study the scaling properties of our approach with larger datasets. Our experiments were conducted on a laptop with 2.3GHz Intel Core i5 processor and 16GB RAM and the implementation used Python v3.6.3 packages.

4.6 Discussion

a. What applications would benefit from multi-modal hierarchical clustering? Apart from being an interesting theoretical problem, we can think of several real-world applications that could benefit greatly from a hierarchical decomposition. Based on our expertise and interest, we consider the dynamics of an online community which includes many hot-button applications such as online fraud, fake news dissemination, online review tampering, and opinion manipulation, as we explain below. As online seems to dominate physical interactions, there is a plethora of online communities, from social networks to discussion forums, and collaboration platforms. Analyzing these communities can provide immensely useful information on who interacts with whom and for what purpose. Specifically, one may want to a) identify groups of misbehaving agents, including cyberbullying and harassment, b) detect online collaboration and collusion, c) detect information spread over time. We argue that any such analysis can greatly benefit by having a capability to automatically provide the interaction landscape through a hierarchy of clusters.

More generally, there is a plethora of applications with multi-modal datasets in the financial, and medical sectors that could benefit from an effective multi-modal hierarchical analysis. For example, one could consider the study of drug efficacy over a long period of time, where grouping patients by medical, demographic, and behavioral features, as they evolve over time could illustrate the effects of the drug better. In the *Related Work* section, we provide more examples of such applications.

b. Does RecTen introduce artificial hierarchical structure in the absence of such? A valid concern is whether the perturbation generates hierarchical structure that is not there in the initial data. Although possible, but we argue that under the right conditions, it does not. First, the end-users can control the amount of perturbation, and keeping it reasonably low can minimize the danger. Second, our experimental results on real data suggest that even large clusters are fairly resilient to perturbation for a wide range of perturbations if there is no sub-cluster in those. Indicatively, we can refer to our analysis on the MPGH forum. We find that the biggest cluster (450 users, 530 threads, 23 weeks) discusses about general topics on gaming strategies. We vary the Deletion Percentage, ϵ, from 2% to 15%. For all values, the large cluster was nearly the same each time, and it was never pushed to further decomposition even for high values of the parameter. In fact, we have experimental indication that RecTen produces almost same hierarchy as ground truth for hierarchical data and respects the flatness of non-hierarchical data as well. For instance, experimenting with $k = 15, \epsilon = 6$, and $\lambda = 0.8$ in the non-hierarchical synthetic data, RecTen leads to an average weighted level of 1.14, which is very close to the ideal value of 1 for this case.

However, in a similar vein, one can question if the perturbation introduces information loss. The answer here depends on how one will use the hierarchical clustering. If it will be used to compress the information, then some information loss will take place. However, if the clustering will only be used to infer an underlying structure (e.g. assign nodes to clusters), the question of information loss becomes less relevant compared to the question as to whether the hierarchical clustering is meaningful. The two arguments we would like to make here that would hopefully close this case are: (i) the deletions are meant to disentangle the lower-level clusters which have enough cross-cutting connections that the top-level decomposition deemed to be a single cluster, (ii) the decomposition is known (as it did in the top level) to impute missing values that are needed in order to represent a cluster as a rank-one entity. Therefore, even if the deletions go a bit further than needed (i.e., start deleting more intra-cluster and less inter-cluster connections), the rank-one modeling is able to complete those deletions, provided that sub-clusters are present and deletion is not extensive.

c. Is our evaluation sufficient given the absence of extensive ground truth? We would have loved to have tested our algorithm against a well-established benchmark. Given its absence, we followed a two-prong approach. First, we evaluate RecTen with synthetic data, where we can know the ground truth, and create a wide range of datasets. In addition, we compare with six different state-of-the-art methods. Second, we resort to manual inspection of our analysis on real datasets. We argue that our evaluation provides sufficient evidence of the overall effectiveness and competitiveness of our method. In the future, we will evaluate our work on more synthetic and real datasets.

4.7 Related Work

To the best of our knowledge, RecTen is the first tensor-based approach that extracts a multilevel hierarchical clustering from multi-modal data recursively. We highlight the most relevant and recent methods, which we group into the following categories.

a. Discovering hierarchical structures without using tensor decomposition. Hierarchical structure discovery is a very common task in data mining. The algorithms that are being used mostly include but not limited to bottom-up Agglomerative Hierarchical Clustering, top-down Hierarchical k-means Clustering, and variations of these algorithms.

The basic and widely used version of Agglomerative Hierarchical Clustering is Ward's method, AHC_ward [179]. Different variations of bottom-up Hierarchical Clustering are being used recently as well [165, 110]. DLORE-DP [31] focuses on developing a local cores-based hierarchical algorithm for dataset with complex structures. Another work, AHC_freq [109], proposes an improved frequency-based agglomerative clustering algorithm for detecting distinct clusters on two-dimensional dataset. A recent minimum spanning tree-based bottom-up hierarchical clustering is Affinity [11] which works well for extracting structures from graphs. Note that the above-mentioned methods do not applicable for multi-modal data.

Variations of Top-down hierarchical algorithm like hierarchical k-means is also being used in different domains ranging from modeling blast-produced vibration [124] to large graph embedding [126]. DenPEHC [187] presents a top-down density peak-based

hierarchical clustering method which introduces a grid granulation framework to enable Den-PEHC extract clusters from large-scale and high-dimensional datasets. Recent methods [147, 26] utilizes the sparsest cut to extract hierarchical clusters for the data. [93] focuses on hierarchical document clustering leveraging non-negative matrix factorization. All the above-mentioned algorithms suffer from the same problem. These algorithms are applied on data represented in 2D matrix format. Therefore, finding clusters in multi-modal data requires new strategy. The work of [60] focuses on tri-clustering in time-evolving graphs but did not utilize tensor factorization at all.

Different variations of deep learning-based clustering strategies are also prominent. But they basically use the deep neural networks to generate the features and then apply traditional machine learning clustering algorithms to finally compute the clusters [87, 168]. However, none of these strategies have ever been applied on multi-modal hierarchical clustering.

b. Advanced tensor decomposition: interactivity and hierarchical data. Although the recursive use of tensor is very rare, a very recent work [4] uses a two-level tensor decomposition to detect fake news. Though the authors use the term "hierarchical" for their model, it is only two-levels and combines disparate datsets to achieve its goal. Another work [178] proposes an interactive framework using tensor decomposition in order to detect topic hierarchy, which differs from the recursive tensor factorization that we do in this study.

c. Tensor decomposition approaches and applications. Tensor decomposition is a well-studied area of research. For our work, we have used CP decomposition but

there are other bunch of tensor decomposition approaches. Tucker Decomposition [88] is the most well-known of them but it is not capable of generating unique decomposition. There are other tensor clustering approaches but they are applicable in focused domain, for example, tensor graph clustering to detect higher-order cycles [15], approximation algorithm for 1-d clustering [80]. Another recent tensor-based clustering is Dynamic Tensor Clustering (DynamicT) [163] which works better for dynamic tensors but struggles for general tensors. All of the above-mentioned algorithms suffer from the same general problem of not being hierarchical.

Tensor decomposition has a wide range of applications in diverse domains for categorical data [74, 91, 107, 130].

Relatively recently tensor-based techniques have been used in social media analysis. Very recent approaches, TenFor and HackerScope [74, 76, 72, 73], use non-hierarchical tensor decomposition to find interesting events and hackers' dynamics in social media and forums. TimeCrunch [155] focuses on mining some temporal patterns from time-evolving dynamic graphs. More recent studies [130] use tensor to model multilingual social networks in online immigrant communities. Other works [107, 61] use tensor decomposition to study the online communities and their evolution.

4.8 Conclusion

We propose, RecTen, an unsupervised tensor-based approach to systematically discover hierarchical clusters in a multi-dimensional space. It can operate parameter-free with default values, but optionally allow parameter tuning for an expert end-user. We show the effec-

tiveness of our approach by an extensive evaluation using both synthetic data and five real datasets.

From an algorithmic point of view, the key advantages of our approach could be summarized in the following points: a) we harness the power of tensor decomposition, b) we provide hierarchies, and (c) we compare favorably to or outperform previous methods. From a practical point of view, RecTen has three attractive features: (a) it operates in an unsupervised way, (b) it generalizes well to both categorical and numerical multi-modal data, and (c) it can operate with default parameters, or customized by a savvy user.

Our work is a step towards a powerful capability, which can allow the data analysts and researchers to mine the wealth of information that exists in massive multi-modal data. Our commitment to providing our tools and data to researchers and practitioners will hopefully amplify the impact of this work.

4.9 Acknowledgements

This work was supported by the UC Multicampus-National Lab Collaborative Research and Training (UCNLCRT) award #LFR18548554.

Chapter 5

HackerScope: The Dynamics of a Massive Hacker Online Ecosystem

5.1 Introduction

"How can a 17 year old kid from Florida [2] be reportedly the mastermind behind the recent hacking of Twitter? This is the motivational question behind this work.

The security community has limited understanding about their "enemy" because of limited understanding of malicious hackers and their interactions. The hacker community is fairly wide encompassing curious teenagers, aspiring hackers, and professional criminals. However, the hackers are surprisingly bold in leaving a digital footprint, if one looks at the right places in the Internet. Sometimes they boast of their successes and share hacking-related information in online platforms.

How can we begin to understand the ecosystem of malicious hackers based on their online footprint? The input is the online activities of these hackers, and the goal is to answer the following questions: (a) do these hackers work in groups or alone, and (b) who are the most influential hackers? Here, we consider two types of platforms that hackers frequent: (a) software archives, and (b) online security forums. Popular and public software archives, such as GitHub provide shelter for **malware authors**, who create publicly-accessible malware repositories [144]. Furthermore, online forums have recently emerged as marketplaces and information hubs of malicious activities [52, 137]. In the rest of this paper, we will use the term **hacker** to refer to actors who develop and use software of malicious intent. We will also use the term *hackers* and *malware authors* interchangeably, although some malware authors may not have malicious intent.

There is limited work for the problem as defined above. First, we are not aware of a study that systematically profiles the dynamics of the online hacker ecosystem, and especially one considering software archives. Most of the previous efforts on GitHub follow a software-centric view or study GitHub at large without focusing on malware [23] [22] [19]. Most of the previous works on online forums focus on identifying emerging topics and threats [52, 137]. Other efforts report malware activity, focusing on hacking events, and much less, if at all, on the ecosystem of hackers [149, 151]. We elaborate on previous works in Section 5.8.

We propose **HackerScope**, a systematic approach for modeling the ecosystem of malware authors by analyzing their online footprint. We start with an extensive analysis

of malware authors on GitHub, as this is a significantly less-studied space. We then use security forums to find more information about these authors. From an algorithmic point of view, we use three network representations: (a) the author-author network, (b) the author-repository network, and (c) cross-platform egonets, which we explain later. In addition, we use some basic Natural Language Processing techniques, which we intend to develop further in the future.

We apply and evaluate our approach using 7389 malware authors on GitHub over the span of 11 years and leverage the activity on four security forums in the grey area between white-hat and black-hat security. GitHub is arguably the largest repository with roughly 30 million public repositories, while, appropriately fine-tuned, our approach can be used on other software archives. Our approach encompasses four research thrusts, which identify and model: (a) statistics and trends, (b) communities of hackers and their dynamics, (c) influential hackers, and (d) hacker profiles across different online platforms. For the latter type, we show the collaborators of hackers as captured by the cross-platform egonets spanning GitHub and security forums in Fig. 5.1. Our key results are summarized in the following points.

a. The ecosystem is growing at an accelerating rate: The number of new malware authors on GitHub is roughly tripling every two years. This alarming trend points to the importance of monitoring this ecosystem.

b. The ecosystem is highly collaborative: We find 513 collaboration communities on GitHub with high cohesiveness (*Modularity Score* within [0.65-0.78]), including many large communities with hundreds of users. The malware community is very collab-

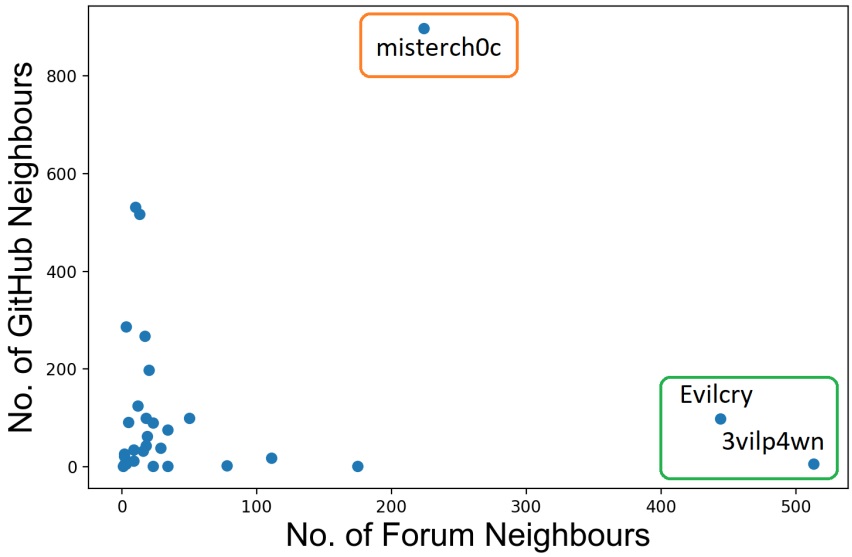


Figure 5.1: Profiling hackers across platforms using our cross-platform egonet: the scatter-plot of the number of neighbors on GitHub versus those on security forums for 30 malware authors as captured in our cross-platform egonet.

orative: a malware repository is forked *four times* more compared to a regular GitHub repository.

c. We identify a group of 1.7% of influential authors: We develop a systematic approach to determine the influence among malware authors. Our novelty lies in: (a) considering many types of interactions, and (b) capturing the network-wide influence of an author. We find a core group of 1.7% of the malware authors, who are responsible for: (a) generating influential repositories, and (b) providing the social backbone of the malware community.

d. We identify professional hackers in the ecosystem: We find that 30 authors are professional *malicious* hackers. Going across platforms, we find GitHub authors who are quite active on our security forums. We show the evidence that these are pro-

fessional hackers, who are building an online "brand". For example, user *3vilp4wn* is the author of a keylogger repository on GitHub, which he promotes in the *HackThisSite* forum using the same username (shown at bottom right in Fig. 5.1).

Our work in perspective. The proposed work is part of an ambitious goal: we want to model the Internet hacker ecosystem at large as it manifests itself across platforms. Our initial results are promising: a) the hackers seem to want to establish a brand, hence they want to be visible, and b) a cross-platform study is possible, as some authors maintain the same login name. Our systematic approach here constitutes a building block towards the ultimate goal. With appropriate follow up work, achieving this goal can have a huge practical impact: security analysts could prepare for emerging threats, anticipate malicious activity, and identify their perpetrators.

Open-sourcing for maximal impact. We use Python v3.6.2 packages to implement all the modules of HackerScope. We intend to make our datasets and tools public for research purposes.

5.2 Background and Data

Our work focuses on GitHub, the largest software archive with roughly 30 million public repositories, and uses data from security forums. Although GitHub policies do not allow malware, authors do not seem to abide by them. A brief description of our dataset is presented in Chapter 2.

5.3 Our Approach

We have an ambitious vision for our approach, which we plan to release as a software platform. We provide a brief overview in Fig. 5.2. In this paper, we will elaborate on the four analysis modules: (a) a statistics and trends module, which provides the landscape of primary behaviors of the ecosystem (Section 5.4), (b) a community analysis module, which identifies and profiles communities of collaboration (Section 5.6), (c) an influence analysis module, which defines and calculates the significance of authors (Section 5.5), and (d) cross-platform analysis module (Section 5.7).

In addition, our approach also includes: a data collection module, which aggregates, cleans and preprocesses the raw information; a control center module; and a reporting module. These modules are not equally developed, while at the same time, we could not provide all the types of results that we have available due to space limitations.

Below, we highlight some interesting or novel aspects of our approach, which are often cutting across several modules.

a. Synthesizing multi-source data. Our approach focuses on data for authors from GitHub and combines it with additional data from security forums, and Internet searches.

b. Defining appropriate features. As we already saw, the authors and the repositories have a very rich set of interactions. We have primary (measured directly) and secondary (derived from the primary) features, which need to be determined carefully to capture effectively the dynamics of the ecosystem.

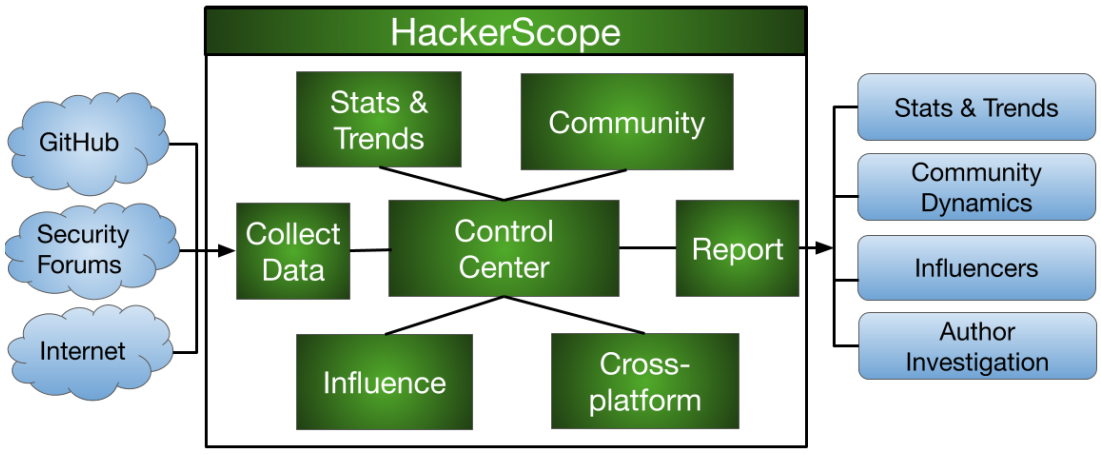


Figure 5.2: The overview of our approach highlighting the key functions.

c. Modeling the dynamics. We use three network representations to capture the rich interactions and relationships among authors and repositories. The network representations include: (a) the author-author network, (b) the author-repository network, and (c) cross-platform egonets.

d. Reporting behaviors. The goal is to provide intuitive and actionable information in an appealing and ideally interactive fashion. The results in this paper provide an indication of some initial plots and tables that our approach will provide to the end user, who could be a researcher or a security analyst.

5.4 Statistics and Trends

This section describes the functionality of the *statistics and trends* module of our approach, whose intention is to provide a basic understanding of author behaviors.

A. Basic distributions of malware authors. We study the complementary cumulative distribution function (CCDF) of three metrics: (a) the number of repositories created, (b) the number of followers, and (c) sum of the number of forks across all the malware repositories of the author. As expected all distributions are skewed, but the plots are omitted due to space constraints. First, we find that 15 authors are contributing roughly 5% of all malware repositories, while 99% of all authors have created less than 5 repositories each. Second, we find that 3% (221) of the authors have more than 300 followers each, while 70% of the authors have less than 16 followers. Finally, examining the total number of forks per author, we find that 3% (221) of the authors have their repositories forked more than 150, while 43% of authors encounter at least one fork.

B. Forking behavior: Malware repositories are forked four times more than the average repository. Malware repositories are more aggressively forked, which is an indication of the higher collaboration in the ecosystem. First, we find that a malware repository is forked 4.01 times on average, while a regular GitHub repository is forked 0.9 times, as reported in previous studies [82]. Second, we want to see if this is due to a few popular repositories, but this is not the case. We find that 39% of the malware repositories are forked at least once, while this is true for only 14% for general repositories [82].

C. Trends. *"How fast is this ecosystem growing?"* To answer the question, we plot the number of new malware authors per year in Fig. 5.3. We consider that an author joins the ecosystem at the time that they create their first malware repository in our database.

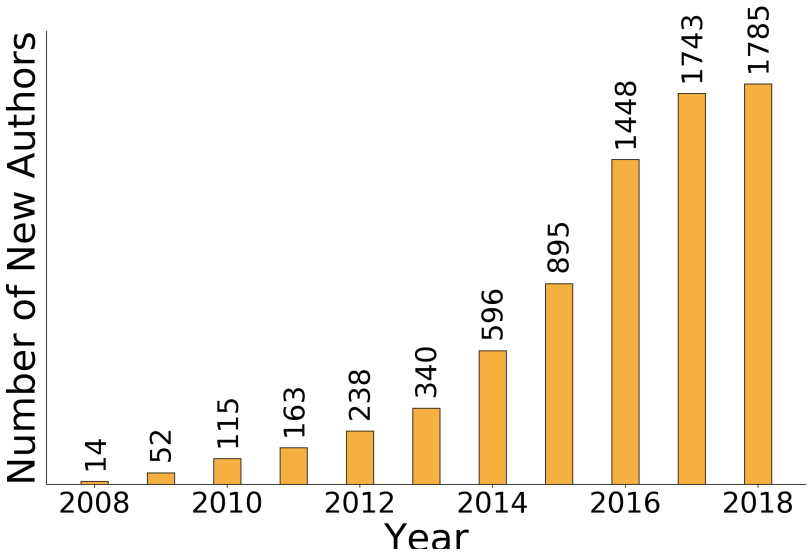


Figure 5.3: New malware authors in the ecosystem per year.

a. The number of new malware authors almost triples every two years. We plot the new malware authors per year in Fig. 5.3. We observe an increase from 238 malware authors in 2012 to 596 authors in 2014 and to 1448 authors in 2016. We also observe a steep 62% increase from 2015 to 2016. This trend is interesting and alarming at the same time.

b. The number of new malware repositories more than triples every four years. Echoing the growth of the authors, the number of repositories is also increasing super-linearly. In the future, we plan to study the trends of malware in terms of both types of malware and its target platform.

5.5 Identifying Influential Authors

To understand the dynamics of the ecosystem, we want to answer the following question: *"Who are the most influential authors?"* The functionality in this section is part of the *influence analysis* module of Fig. 5.2.

A. HackerScore: Identifying influential authors. We argue that finding influential authors presents several challenges. First, there are many different activities and interactions, such as creating repositories, commenting, following other authors and being followed by other authors. Second, we can consider two types of actions: (a) creating influential artifacts, (b) observing and engaging with other people and artifacts. Furthermore, the distinction is not always clear. For example, forking a repository creates a new, but derivative, repository.

To address the above challenges, we take **socially-aware approach** to influence: creating a few influential repositories is more important than creating many non-influential repositories. We discuss how we model and calculate this influence below.

The Author-Author graph (AA). We create the Author-Author network to capture the network-wide interaction among authors. We define a weighted labeled multi-digraph: $G(V, E, W, L_e)$ where V is the malware author set, E is the set of edges, W is the weight set and L_e is the set of labels that an edge e can be associated with. These labels correspond to different types of relationships between authors. Here we opted to consider only malware authors in the graph to raise the bar for being part of the hacker community.

The types of interactions. We consider four types of relationships between authors here. A directed edge (u, v) from author u to v can be (i) a follower edge: when u follows v, (ii) a fork edge: when u forks a repository of v, (iii) a contribution edge: u contributes code in a repository of v, and (iv) a comment edge: u comments in a repository of v. These relationships capture the most substantial author-level interactions.

The multi-graph challenge and weight calibration. Our graph consists of different types of edges, which represent different relationships that we want to consider in tandem. The challenge is that the relationships have significantly different distributions, which can give an unfair advantage or eliminate the importance of a relationship. For example, contribution activities are rarer compared to following, but one can argue that a contribution to a repository is a more meaningful relationship and it should be given appropriate weight.

For fairness, we make the weight of a type of edge inversely proportional to a measure of its relative frequency. In detail, we calculate the average degree d_{type} for each *type* of edge: follower, fork, contribution, and comment from the subgraph containing only that type of edges from the AA graph. We find the following average degrees: $d_{follower} = 12.21$, $d_{fork} = 4.67$, $d_{contribution} = 0.53$ and $d_{comment} = 0.49$. We normalize these average degrees using the minimum average degree ($d_{min} = 0.49$) and we get the *inverse* of this value as the weight for that edge, namely, d_{min}/d_{type}. This way, we set the following weights: $w = 0.04$ for a following edge, $w = 0.1$ for a forking edge, and $w = 1$ for a commenting or a contribution edge. This enables us to consider each relationship type more fairly.

We propose a socially-aware and integrated approach to combine all the author activities in a single framework. First, we identify and define two roles in the ecosystem: (a) **producers**, who create influential malware repositories, and (b) **connectors**, who enhance the community by engaging with influential malware authors and repositories. To calculate the roles of the malware authors, we first model the interaction among authors in the AA graph described above. Next, we apply our algorithm, a customized version of a weighted hyperlink-induced topic search algorithm modified to handle the multiple types of relationships between authors. We discuss the related algorithms in Section 5.8.

Calculating the HackerScore. We associate each node u with two values: (a) a **Producer HackerScore** value, PHS_u, and (b) **Connector HackerScore** value, CHS_u. Let $w(u, v)$ be the weight of edge (u, v) based on its label, as discussed above.

The algorithm iterative refines the producer and connector values until it converges. We, now, elaborate on the steps. First, PHS_u and CHS_u are initialized to 1. During the iterative step, the algorithm updates the values as follows: (i) for all v pointing to u: $PHS_u = \sum_v w(v, u) * CHS_v$, or zero in the absence of such edges, (ii) for all z pointed by u: $CHS_u = \sum_z w(u, z) * PHS_z$, or zero in the absence of such edges, and (iii) we normalize PHS_u and CHS_u, so that $\sum_u PHS_u = \sum_u CHS_u = 1$. For the convergence, we set a tolerance threshold of 10^{-9} for the change of the value of any node. After 449 iterations, we obtain the two HackerScore values for each author.

Identifying influential malware authors. In Fig. 5.4, we plot the Connector HackerScore versus Producer HackerScore for our malware authors. Separately, we identify "knees" in the individual distributions of each score at PHS = 0.00215 and CHS =

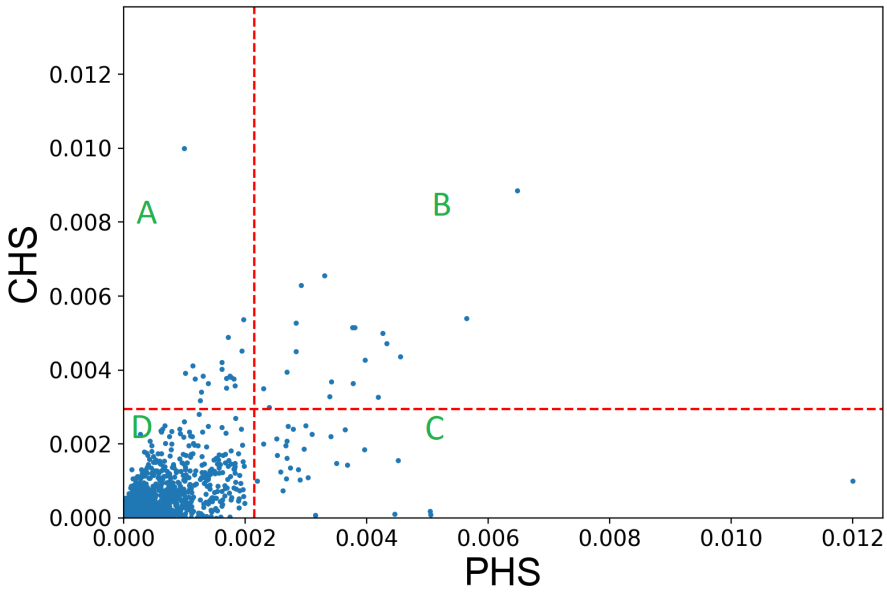


Figure 5.4: The scatterplot of the Connector HackerScore vs. Producer HackerScore for the malware authors in our GitHub dataset.

0.0029 indicated by the red dotted lines. This way, we observe four regions defined by the combination of low and high values for PHS and CHS values which shows if an author is influential as producer or connector.

A few authors (1.7%) drive the community. The three regions of influence together consist of 128 malware authors (1.7%). The break down of the region size is fairly even: Region A of mostly connector authors devoted to connect the malware community is 0.6%, Region C of the influential producers who are the originator of the malware resources is 0.7%, and Region B of dual influence is 0.4%. We use the term **Highly Influential Group (HIG)** to refer to this group of authors.

We provide a profile overview of the two most influential authors per region in Table 5.1. The most influential author of Region C is *cyberthrets*, with the highest PHS (0.012)

Name	PHS	CHS	Repos	Followers	Forks	Comments	Contributors
cyberthrets	**0.012**	0.001	336	1013	778	13	2
ytisf	**0.005**	10^{-6}	12	606	1412	4	1
critics	0.001	**0.01**	6	396	83	446	301
samyk	0.0018	**0.006**	2	554	125	176	209
D4Vince	0.007	**0.008**	7	608	499	165	187
n1nj4sec	**0.006**	0.005	8	876	1391	64	79

Table 5.1: The profiles of the two most influential malware authors from each region A, B, and C.

and 336 malware repositories. She gained a huge following by creating all her repositories of assembly code malware on Feb 16, 2016. The top connector author from Region A is *critics* with a CHS score of 0.01, which stems from her 446 comments across 301 repositories. The top malware author from Region B is *D4Vince* for his dual role in producing credential reuse tools with 7 repositories and 165 comments and 187 contributions.

The importance of socially-aware significance. We argue that our socially-aware definition of significance provides more meaningful results than simply taking the top-ranked users in any primary metric in isolation. First, the two scores capture different aspects of influence: they can differ by orders of magnitude as is the case with *cyberthrets* and *ytisf*. Second, our scores capture a combined network-wide influence that each primary metric could miss. For example, our most influential producers do not always own many malware repositories. Malware author *D4vince* and *n1nj4sec*, mentioned in Table 5.1, have single-digit repositories (7 and 8 respectively) and yet are two of the top producers. On the other hand, author *kaist-is521* is ranked way below than *n1nj4sec* in terms of HackerScore (PHS =0.0001 and CHS =0.00013), although she has 18 malware repositories.

B. Reciprocity of interactions. We want to understand better the nature of the author interactions here.

"Is the influence among malware authors reciprocal?" The answer is negative: **the relationships are not reciprocal**, which is in stark contrast to the reciprocal relationships in other social media like Twitter and Facebook [181]. We consider a total of six relationships: following, forking, commenting, contributing, watching, and starring relationships. We define the **Reciprocity Index** for relationship x, RI_x, to be the ratio of reciprocal relationships over the pairs of authors with that type of relationship (unilateral or mutual) in the Author-Author network.

We find that the reciprocity is low and less than 7.3% for all the relationships in question. By contrast, reciprocity is often above 70% in social media, like Facebook or Twitter [181]. These social media mirror personal relationships and have an etiquette of conduct. We conjecture that the lower reciprocity on GitHub is due to its utilitarian orientation: following an author stems from a professional interest.

5.6 Community Analysis

This section describes the functionality of the *community analysis* module, whose goal is to identify the communities of collaboration among the malware authors on GitHub.

A. Identifying collaboration communities. We quantify the collaborative nature of the malware authors as follows.

The Author-Repository graph (AR). We define the Author-Repository graph to be an undirected bipartite graph, $G = (A, R, E)$, where A is the set of malware authors and R is the set of malware repositories. An edge $(u, r) \in E$ exists, if author u: (a) creates, (b) stars, (c) forks, (d) watches, (e) comments, or (f) contributes to repository r.

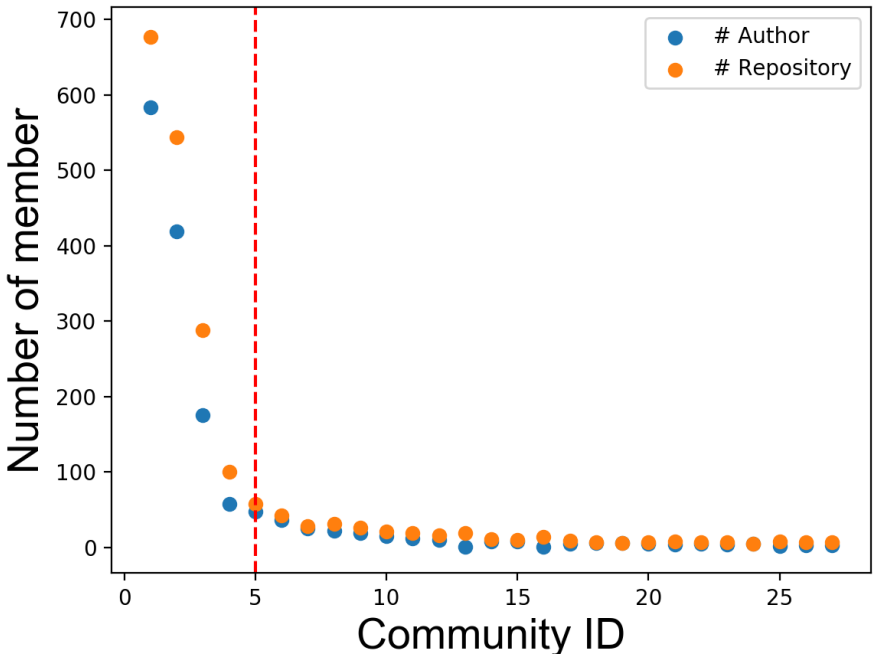


Figure 5.5: The distribution of the number of authors and repositories for the 27 largest communities in the order of community size.

Identifying bipartite communities. To identify communities, we employ a greedy modularity maximization algorithm modified for bipartite graphs as we discuss in our related work.

We find a total of 513 communities spanning a wide range of sizes as shown in Fig. 5.5. The size of the communities follows skewed distribution. In Fig. 5.5, we plot the number of malware authors and repositories per community in order of decreasing community size (defined as the sum of authors and repositories). We find that 90% of communities have less than 14 authors and repositories. We also see a fairly sharp knee in the plot at the fifth community, as shown by the vertical line.

B. Profiling the communities. A full investigation of the purpose, evolution, and internal structure of each community could be a research topic in its own right. Here, we only provide an initial investigation around the following three questions.

a. How cohesive are our communities? We report the **Modularity Score** ***(MS_C)***, which quantifies the cohesiveness of a community C. The MS_C is defined as follows: $MS_C = \frac{n_C(E)}{N_C(E)}$, where $n_C(E)$ is the total number of edges and $N_C(E)$ is the number of all possible edges in community C (if the community was a bipartite clique).

Overall, **our communities are highly cohesive**: 82.8% (425) of the communities have a *Modularity Score* $MS_C \geq 50\%$, which means that more than half of all possible edges within the community exist. Interestingly, the largest communities exhibit strong cohesiveness. In Table 5.2, we present a high-level profile of the five largest communities which have a Modularity Score of 0.65-0.78, which is indicative of tightly-connected communities.

b. Who are the community leaders? We want to identify the influential authors as part of profiling a community. We identify the top two most influential producers and connectors per community using the HackerScore from Section 5.5. This leads us to a group of 144 leaders of the communities of size of at least 20 authors. We find 81% of these community leaders are part of the Highly Influential Group (HIG) of authors. This suggests that the HIG authors are indeed driving forces for the ecosystem. In the future, we intend to investigate in more depth the influence and dynamics of each community.

c. What is the focus of each community in terms of platform and malware type? A security expert would want to know the main type of malware (e.g. ransomware) and

the target platform (e.g. Linux) of a community. We use the Repository Keyword Set, W_r, information of a repository r, as we defined in Section 5.2, and we use it to characterize the community.

One way to quantify the importance of a keyword for a community is to measure the number of repositories, for which that keyword appears at least once. In detail, we use the **Strength Of Presence** *(SOP)* metric, which we define as follows. For a community C with a set of R repositories, we define k_i to be the number of repositories, in which keyword i appears in the metadata W_r for repository r at least once for all repositories $r \in R$. We define the SOP_i of keyword i from keyword set S as follows: $SOP_i = \frac{k_i}{\sum_{j \in S} k_j}$. In Table 5.2, we show the most dominant keywords from malware types and platforms sets for each community and the related SOP scores.

We can also use the *SOP* to visualize the keywords as a word-cloud. A word-cloud is a more immediate, appealing, and visceral way to display the information. In Fig. 5.6, we show the word-cloud for the third largest community, which is dominated by *ransomware* malware and targets *Windows* platforms. Not only we see the main words stand out, but their relative size conveys their dominance over the other words more viscerally than a lengthy table of numbers.

We present the results of this type of profiling for the largest communities in Table 5.2, which we also discuss below.

We find that the largest community of 584 malware authors and 677 malware repositories having Linux ($SOP = 0.32$) and keylogger ($SOP = 0.29$) as the dominant platform and malware type. Interestingly, we find that 49 of the top 100 most prolific (in

ID	Authors	Repos	MS	Dominant Platform	SOP	Dominant type	SOP
1	584	677	0.65	Linux	0.32	Keylogger	0.29
2	419	544	0.67	Windows	0.26	Virus	0.31
3	175	288	0.73	Windows	0.65	Ransomware	0.44
4	57	100	0.78	Linux	0.43	Spyware	0.43
5	47	57	0.71	Mac	0.33	Trojan	0.22

Table 5.2: High-level profile of the five largest communities of malware authors and malware repositories.

Figure 5.6: The word-cloud for the malware types and platforms keywords for the third largest community: Ransomware and Windows dominate.

terms of the number of repositories created) authors are in this community. Upon closer investigation, we find that 11 out of the 15 authors with the highest degree in the subgraph of this community are keylogger developers.

The third-largest community consists of 175 malware authors and 288 malware repositories and revolves around Ransomware ($SOP = 0.65$) and Windows platform ($SOP = 0.44$). For reference, we present the word-cloud of the malware types and platforms based on the SOP score in Fig. 5.6 for this community which exhibits that Ransomware and Windows possess the highest SOP scores.

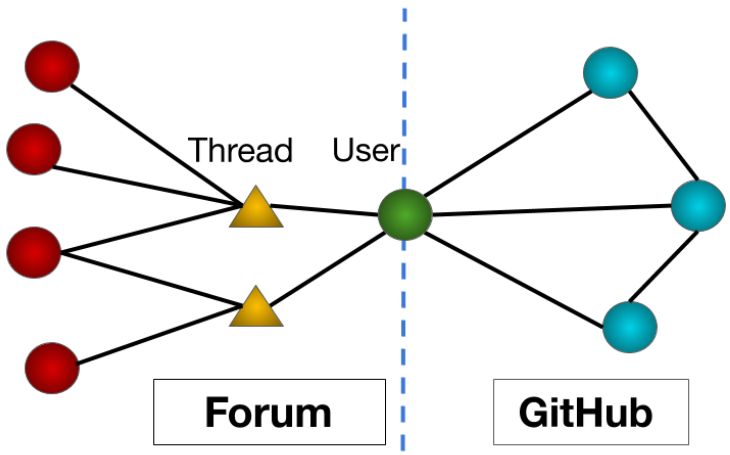


Figure 5.7: A cross-platform egonet: capturing the neighbors of both the security forum and GitHub.

Finally, the fourth largest community (57 authors, 100 repositories) is the most tightly connected ($MS = 0.78$) and it revolves around the development of attack tools for Kali Linux. Upon closer inspection, we find that 15 of the top 25 authors (based on node degrees) form an approximate bipartite clique with 5 repositories. This group developed *WiFiPhisher* in 2016, a Linux-based python phishing tool [162], which has been used for both good and evil [37].

The above are indicative of the potential information that we could extract from these malware repositories. In the future, we intend to: (a) extract more detailed textual information from each community, and (b) study the evolution and dynamics of these communities over time.

Name	Posts in forum	Collaborators in forums	Malw. repo	Followers	Collaborators in GitHub	Repository content	Internet-wide Reputation
misterch0c	7	224	7	749	898	Cracked malware code	Self-declared hacker
3vilp4wn	103	513	1	0	6	Python keylogger	Keylogger developer
fahimmagsi	73	175	1	1	1	Backdoor	Famous hacker
Evilcry	18	444	2	89	98	Botnet and ransomware	Ransomware expert

Table 5.3: Profiles of four cross-platform users from WS, HTS, OC and EH forum respectively.

5.7 Author Investigation

"Who are these malware authors?" To answer this question, we go across platforms to security forums and leverage our datasets from several security forums. The functions described here are part of the *author investigation* module of Fig. 5.2.

a. Malware authors strive for an online "brand" and usernames seem persistent across online platforms. We find that many malware authors use the same username consistently across many online platforms, such as security forums, possibly in pursuit of a reputation.

We identify 30 malware authors who are active in one of our four security forums: 12 in Wilders Security, 6 in Ethical Hacker, 4 in Offensive Community, and 8 in Hack This Site [154]. We argue that some of these usernames correspond to the same users based on the following two observations.

First, we find significant overlap in the interests of the cross-platform usernames. For example, usernames *int3grate* and *jedisct1* show interest in ransomware in both platforms, while *3vilp4wn* advertises her keylogger malware (github.com/ 3vilp4wn/CryptLog)

in the forum. Second, these usernames are fairly uncommon, which increases the likelihood of belonging to the same person. For example, the top ten results from internet searching for the username of author *Misterch0c* returns nine hacker related sites and a twitter account with a different handle but claimed by Misterch0c. Note that not all the malware authors or repositories have a malicious purpose. For instance, the project "Empire" [45] by *xorrior* was created as an offensive tool to stress-test the security of systems. However, it has recently been used by the state-sponsored hacking group *Deep Panda* [119]. In general, offensive security tools contribute to the power of the malware ecosystem irrespective of the intention of its creator.

b. Modeling the cross-platform interactions. We propose to study the cross-platform interactions between GitHub and security forums as a promising research direction that can bridge two domains: software repositories and online forums.

We define the **cross-platform egonet** of a user as one that consists of her egonets from the two platforms as shown in Fig. 5.7. The forum egonet captures the interaction of the users that post on the same threads, while we leverage the Author-Author network to define the GitHub egonet.

The value of cross-platform analysis. Using the cross-platform egonet as a basis, we can model the cross-platforms user dynamics, and more specifically, we can: (a) identify common "friends" between the ego-nets, (b) find the topics of interest and activities in each egonet, and (c) model information flow and influences across platforms. In Fig. 5.1, we visualize the activity of a cross-platform user by comparing the number of users on each side of the egonet as shown in Fig. 5.7.

In Table 5.3, we show the actual values of indicative users, including the three outliers in the plot.

The cross-platform egonet analysis can enrich the profile of each user significantly. For example, if we were just looking at GitHub, we may not have paid attention to *3vilp4wn* and *Evilcry*. Both of these authors are less active on GitHub (small GitHub egonet), but are quite active in the security forums (large forum egonet). A closer investigation of the security forums reveals activities that match their interests on GitHub. This suggests that their GitHub activity is part of their online brand. For example, *3vilp4wn* advertises her GitHub keylogger repository in the forum.

c. Using information from the web. In our approach, we leverage existing information on hackers from (a) security outlets and databases, and (b) using web queries. With our python-based query and analysis tools, we verified the role and activities of authors, which we omit due to space limitations.

5.8 Related Works

Studying the dynamics of the malware ecosystem on GitHub has received very little attention. Most studies differ from our work in that: (a) they do not focus on malware on GitHub, and (b) when they do, they do not take an author-centric angle as we do here: they focus on classifying malware repositories or use a small set for a particular research study.

Our work builds on our earlier effort [144, 146], whose main goal is to identify malware repositories on GitHub at scale, but it does not study the malware author ecosystem as we do here.

a. Studies of malware repositories on GitHub: Several other efforts have manually collected a small number of malware repositories with the purposes of a research study [101, 197, 146]. Some other studies [23, 22, 84] analyze malware source code from a software engineering perspective, but use only a small number of GitHub repositories as a reference.

b. Studies of benign repositories on GitHub: Many studies analyze benign repositories on GitHub from a point of view of software engineering or as a social network. Some efforts find influential users and analyze the motivation behind following, forking, and contributions [19, 82]. Earlier efforts study repositories by analyzing the repository-repository relationship graph [167], and by using an activity graph [185].

Several works in this area identify influential authors and repositories using: the starring activity [71], the Following-Star-Fork activity [70], or a rank-based approach [105]. Note that a version of the hyperlink-induced topic search algorithm [103] has been used by some of the above efforts for calculating influence, but they do not adjust the weights to account for the different frequencies of the types of interactions between users.

For our bipartite clustering, we adapt the greedy modularity maximization approach [34][6].

c. Studies on security forums: This is a recent and less studied area of research. Most of the works focus on extracting entities of interest in security forums.

An interesting study focuses on the dynamics of the black-market of hacking goods and services and their pricing [137]. Other studies focus on identifying important events and threats [149, 151, 74, 77, 75]. None of the aforementioned works focus on the dynamics among hackers across platforms.

d. Cross-platform study: Finally, some efforts study author activities on different software development forums, namely GitHub and Stack Overflow [64, 97], but do not consider information from security forums.

5.9 Conclusion

We develop a systematic approach for studying the ecosystem of hackers. Our approach develops methods to identify (a) influential hackers, (b) communities of collaborating hackers, and (c) their cross-platform interactions. Our study concludes in three key takeaway messages: (a) the malware ecosystem is substantial and growing rapidly, (b) it is highly collaborative, and (c) it contains professional malicious hackers.

Our initial findings are just the beginning of a promising future effort that can shed light on this online malware author ecosystem, which spans software repositories and security forums. The current work thus can be seen as a building block that can enable new research directions.

5.10 Acknowledgement

This work was supported by the UC Multicampus-National Lab Collaborative Research and Training (UCNLCRT) award #LFR18548554.

Chapter 6

Conclusions

Our thesis proposes and develops a systematic suit of methods to extract actionable information from online platforms. We develop robust tools to (a) mine important "events", (b) facilitate a hierarchical cluster extraction, and (c) model the dynamics of an ecosystem (here, malware authors). Our approaches have the following main advantages: (a) we develop complete tools for each of our methods which provide visual and intuitive information and can be operated even by a savvy users, (b) our tools can operate in unsupervised way without any apriori knowledge, (c) critical hierarchical patterns can also be discovered using one of our methods, and (d) we can track the hackers across platforms and understand their dynamics by profiling them.

Our study concludes in three key takeaway messages: (a) online platforms hide significant amount of security related important information which can be mined systematically and can be used even for early detection of critical events, (b) hackers leave online footprints, collaborate among themselves, brag about their hacking successes even in secu-

rity forums and can be spotted using our tool, (c) malware development are on the rise targetting even macOS and iOS which are thought to be safer. Our initial findings are just the beginning of a promising future effort that can shed light on this online malware author ecosystem, which spans software repositories and security forums.

The current work thus can be seen as a building block that can enable new research directions. Follow-up efforts can use our approach to (a) detect emerging trends, (b) monitor malicious activity, (c) develop new capabilities on top of our methods, and (d) identify influential hackers towards safeguarding the Internet.

Bibliography

[1] 3vilp4wn. Hacking tool of 3vilp4wn. https://github.com/3vilp4wn/CryptLog/, 2013. [Online; accessed 08-February-2020].

[2] Aaron Holmes. 17 years old boy tried to hack twitter. https://www.businessinsider.com/twitter-hacker-florida-teen-past-minecraft-bitcoin-scams-2020-8/, August 2020.

[3] A. Abbasi, W. Li, V. Benjamin, S. Hu, and H. Chen. Descriptive analytics: Examining expert hackers in web forums. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, pages 56–63, Sep. 2014.

[4] Sara Abdali, Neil Shah, and Evangelos E Papalexakis. Hijod: Semi-supervised multi-aspect detection of misinformation using hierarchical joint decomposition. *ECML-PKDD*, 2020.

[5] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 468–471. IEEE, 2016.

[6] Taher Alzahrani and Kathy J Horadam. Community detection in bipartite networks: Algorithms and case studies. In *Complex systems and networks*, pages 25–50. Springer, 2016.

[7] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[8] John Aycock. *Computer viruses and malware*, volume 22. Springer Science & Business Media, 2006.

[9] Ali Sajedi Badashian and Eleni Stroulia. Measuring user influence in github: the million follower fallacy. In *2016 IEEE/ACM 3rd International Workshop on Crowd-Sourcing in Software Engineering (CSI-SE)*, pages 15–21. IEEE, 2016.

[10] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The nepenthes platform: An efficient approach to collect malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

[11] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. In *Advances in Neural Information Processing Systems*, pages 6864–6874, 2017.

[12] Joseph Bayer, Nicole Ellison, Sarita Schoenebeck, Erin Brady, and Emily B Falk. Facebook in context: Measuring emotional responses across time and space. *new media & society*, 20(3):1047–1067, 2018. SAGE.

[13] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE Software*, 30(1):52–66, 2013.

[14] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[15] Austin R Benson, David F Gleich, and Jure Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126. SIAM, 2015.

[16] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[17] Christian Bird, Peter C Rigby, Earl T Barr, David J Hamilton, Daniel M German, and Prem Devanbu. The promises and perils of mining git. In *2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 1–10. IEEE, 2009.

[18] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[19] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. Understanding the popular users: Following, affiliation influence and leadership on github. *Information and Software Technology*, 70:30–39, 2016.

[20] Logsign Blog. Major ransomware events:. https://blog.logsign.com/10-worst-ransomware-attacks-in-the-last-five-years/, March 2018.

[21] Brian Krebs. Spyeye v. zeus rivalry ends in quiet merger. https://krebsonsecurity.com/2010/10/spyeye-v-zeus-rivalry-ends-in-quiet-merger/, 2010. [Online; accessed 14-March-2020].

[22] Alejandro Calleja, Juan Tapiador, and Juan Cabalero. The malsource dataset: Quantifying complexity and code reuse in malware development. *IEEE Transactions on Information Forensics and Security*, 14(12):3175–3190, 2018.

[23] Alejandro Calleja, Juan Tapiador, and Juan Caballero. A look into 30 years of malware development from a software metrics perspective. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 325–345. Springer, 2016.

[24] Charlotte Carter. Romantic scamming in gaming forum. https://www.stuff.co.nz/auckland/106254141/romantic-scammers-preying-on-players-of-online-game-words-with-friends/.

[25] Ferhat Ozgur Catak and Ahmet Faruk Yazı. A benchmark api call dataset for windows pe malware classification. *arXiv preprint arXiv:1905.01999*, 2019.

[26] Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM, 2017.

[27] Fragkiskos Chatziasimidis and Ioannis Stamelos. Data collection and analysis of github repositories and users. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6. IEEE, 2015.

[28] Gengbiao Chen, Zhengwei Qi, Shiqiu Huang, Kangqi Ni, Yudi Zheng, Walter Binder, and Haibing Guan. A refined decompiler to generate c code with high readability. *Software: Practice and Experience*, 43(11):1337–1358, 2013.

[29] Gengbiao Chen, Zhuo Wang, Ruoyu Zhang, Kan Zhou, Shiqiu Huang, Kangqi Ni, and Zhengwei Qi. A novel lightweight virtual machine based decompiler to generate c/c++ code with high readability. *School of Software, Shanghai Jiao Tong University, Shanghai, China*, 11, 2010.

[30] Jingnian Chen, Houkuan Huang, Shengfeng Tian, and Youli Qu. Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3):5432–5435, 2009.

[31] Dongdong Cheng, Sulan Zhang, and Jinlong Huang. Dense members of local cores-based density peaks clustering algorithm. *Knowledge-Based Systems*, page 105454, 2020.

[32] Chris Stobing. ios malwares in 2014. https://www.digitaltrends.com/computing/decrypt-2014-biggest-year-malware-yet/, 2015. [Online; accessed 08-February-2020].

[33] Ashley Claster. News of hacking by vandathegod:. https://www.databreaches.net/dozens-of-government-websites-defaced-by-vandathegod-hacktivists/, March 2019. [accessed March-2020].

[34] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):6, 2004.

[35] Valerio Cosentino, Javier Luis Cánovas Izquierdo, and Jordi Cabot. Findings from github: methods, datasets and limitations. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 137–141. IEEE, 2016.

[36] CR4SH. Hacking tool of cr4sh. https://github.com/Cr4sh/s6_pcie_microblaze/, 2017. [Online; accessed 08-February-2020].

[37] Cybersec. Stealing password in 5 minutes using wifiphisher. https://www.secjuice.com/phishing-with-wifiphisher/, 2018.

[38] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*, pages 1277–1286. ACM, 2012.

[39] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

[40] David Bisson. News of ransomware outbreak. https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/top-10-ransomware-strains-2016/. [Online; accessed 08-February-2020].

[41] Tsuyoshi Deguchi, Katsuhide Takahashi, Hideki Takayasu, and Misako Takayasu. Hubs and authorities in the world trade network using a weighted hits algorithm. *PloS one*, 9(7), 2014.

[42] Lukás Ďurfina, Jakub Křoustek, and Petr Zemek. Psybot malware: A step-by-step decompilation case study. In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pages 449–456. IEEE, 2013.

[43] Lukáš Ďurfina, Jakub Křoustek, Petr Zemek, Dušan Kolář, Tomáš Hruška, Karel Masařík, and Alexander Meduna. Design of a retargetable decompiler for a static platform-independent malware analysis. In *International Conference on Information Security and Assurance*, pages 72–86. Springer, 2011.

[44] EH. Ethical hacker:. https://www.ethicalhacker.net/, March, 2020 2020. [accessed July-2020].

[45] EmpireProject. Project empire. https://github.com/EmpireProject/Empire.

[46] Neil A Ernst, Steve Easterbrook, and John Mylopoulos. Code forking in open-source software: a requirements perspective. *arXiv preprint arXiv:1004.2889*, 2010.

[47] Nicolaas Klaas M Faber, Rasmus Bro, and Philip K Hopke. Recent developments in candecomp/parafac algorithms: a critical review. *Chemometrics and Intelligent Laboratory Systems*, 65(1):119–137, 2003.

[48] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999.

[49] Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619, 1973.

[50] Sri Shaila G, Ahmad Darki, Michalis Faloutsos, Nael Abu-Ghazaleh, and Manu Sridharan. Idapro for iot malware analysis? In *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*, Santa Clara, CA, August 2019. USENIX Association.

[51] Joobin Gharibshah, Evangelos E Papalexakis, and Michalis Faloutsos. RIPEx: Extracting malicious ip addresses from security forums using cross-forum learning. In *PAKDD*. Springer, 2018.

[52] Joobin Gharibshah, Evangelos E Papalexakis, and Michalis Faloutsos. REST: A thread embedding approach for identifying and classifying user-specified information in security forums. *ICWSM*, 2020.

[53] Joobin Gharibshah, Evangelos E Papalexakis, and Michalis Faloutsos. Rest: A thread embedding approach for identifying and classifying user-specified information in security forums. *arXiv preprint arXiv:2001.02660*, 2020.

[54] Joobin Gharibshah, Evangelos E. Papalexakis, and Michalis Faloutsos. Rest: A thread embedding approach for identifying and classifying user-specified information in security forums. *ICWSM'20, Atlanta, GA*, 2020.

[55] GitHub. Repository search for public repositories: Showing 32,107,794 available repository results. https://github.com/search?q=is:public/. [Online; accessed 13-October-2019].

[56] GitHub. Search api v3. https://help.github.com/en/github/searching-for-information-on-github/searching-for-repositories. [Online; accessed 13-October-2019].

[57] GitHub. User search: Showing 34,149,146 available users. https://github.com/search?q=type:usertype=Users/. [Online; accessed 13-October-2019].

[58] Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8(Oct):2265–2295, 2007.

[59] Georgios Gousios and Diomidis Spinellis. Mining software engineering data from github. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 501–502. IEEE, 2017.

[60] Romain Guigoures, Marc Boullé, and Fabrice Rossi. A triclustering approach for time evolving graphs. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 115–122. IEEE, 2012.

[61] Ekta Gujral and Evangelos E Papalexakis. Smacd: Semi-supervised multi-aspect community detection. In *ICDM*, pages 702–710. SIAM, 2018.

[62] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[63] Mahmud Hasan, Mehmet A Orgun, and Rolf Schwitter. Real-time event detection from the twitter data stream using the twitternews+ framework. *Information Processing & Management*, 56:1146, 2019. Elsevier.

[64] Claudia Hauff and Georgios Gousios. Matching github developer profiles to job advertisements. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 362–366. IEEE, 2015.

[65] Richard Healey. Source code extraction via monitoring processing of obfuscated byte code, August 27 2019. US Patent 10,394,554.

[66] Brandon Heller, Eli Marschner, Evan Rosenfeld, and Jeffrey Heer. Visualizing collaboration and influence in the open-source software community. In *Proceedings of the 8th working conference on mining software repositories*, pages 223–226, 2011.

[67] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O Hall, and Adriana Iamnitchi. Mentions of security vulnerabilities on reddit, twitter and github. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 200–207. ACM, 2019.

[68] James Howison and Kevin Crowston. The perils and pitfalls of mining sourceforge. In *MSR*, pages 7–11. IET, 2004.

[69] HTS. Hack this site:. https://www.hackthissite.org/, March, 2020 2020. [accessed July-2020].

[70] Yan Hu, Shanshan Wang, Yizhi Ren, and Kim-Kwang Raymond Choo. User influence analysis for github developer social networks. *Expert Systems with Applications*, 108:108–118, 2018.

[71] Yan Hu, Jun Zhang, Xiaomei Bai, Shuo Yu, and Zhuo Yang. Influence analysis of github repositories. *SpringerPlus*, 5(1):1–19, 2016.

[72] Risul Islam, Md Omar Faruk Rokon, Ahmad Darki, and Michalis Faloutsos. Hackerscope: The dynamics of a massive hacker online ecosystem. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining (ASONAM)*. IEEE/ACM, 2020.

[73] Risul Islam, Md Omar Faruk Rokon, Ahmad Darki, and Michalis Faloutsos. Hackerscope: The dynamics of a massive hacker online ecosystem. *Social Network Analysis and Mining*, 11(1):1–12, 2021.

[74] Risul Islam, Md Omar Faruk Rokon, Evangelos E. Papalexakis, and Michalis Faloutsos. Tenfor: A tensor-based tool to extract interestingevents from security forums. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining (ASONAM)*. IEEE/ACM, 2020.

[75] Risul Islam, Md Omar Faruk Rokon, Evangelos E. Papalexakis, and Michalis Faloutsos. Recten: A recursive hierarchical low rank tensor factorization method to discover hierarchical patterns in multi-modal data. In *Proceedings of the International AAAI Conference on Web and Social Media*, 2021.

[76] Risul Islam, Md Omar Faruk Rokon, Evangelos E. Papalexakis, and Michalis Faloutsos. Tenfor: Tool to mine interesting events from security forums leveraging tensor decomposition. *Lecture Notes in Social Networks*, 2021.

[77] Risul Islam, Ben Treves, Md Omar Faruk Rokon, and Michalis Faloutsos. Linkman: Hyperlink-driven misbehavior detection in online security forums. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining (ASONAM)*. IEEE/ACM, 2021.

[78] Jack T. Youtube video of wifiphisher. https://www.youtube.com/watch?v=pRtxFWJTS4k. [Online; accessed 14-March-2020].

[79] Jack T. Youtube video of wifiphisher. https://www.youtube.com/watch?v=tCwclyurB8I. [Online; accessed 14-March-2020].

[80] Stefanie Jegelka, Suvrit Sra, and Arindam Banerjee. Approximation algorithms for tensor clustering. In *International Conference on Algorithmic Learning Theory*, pages 368–383. Springer, 2009.

[81] James A Jerkins. Motivating a market or regulatory solution to iot insecurity with the mirai botnet code. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–5. IEEE, 2017.

[82] Jing Jiang, David Lo, Jiahuan He, Xin Xia, Pavneet Singh Kochhar, and Li Zhang. Why and how developers fork what from whom in github. *Empirical Software Engineering*, 22(1):547–578, 2017.

[83] Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[84] Mainduddin Ahmad Jonas, Md Shohrab Hossain, Risul Islam, Husnu S Narman, and Mohammed Atiquzzaman. An intelligent system for preventing ssl stripping-based session hijacking attacks. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2019.

[85] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101. ACM, 2014.

[86] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. An in-depth study of the promises and perils of mining github. *Empirical Software Engineering*, 21(5):2035–2071, 2016.

[87] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, and Stefan Decker. Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*, 2020.

[88] Yong-Deok Kim and Seungjin Choi. Nonnegative tucker decomposition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[89] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[90] Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiao-yong Zhou, and XiaoFeng Wang. Effective and efficient malware detection at the end host. In *USENIX security symposium*, volume 4, pages 351–366, 2009.

[91] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[92] Bence Kollanyi. Automation, algorithms, and politics: Where do bots come from? an analysis of bot codes shared on github. *International Journal of Communication*, 10:20, 2016.

[93] Da Kuang and Haesun Park. Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 739–747, 2013.

[94] Sanjeev Kumar, Rakesh Sehgal, and JS Bhatia. Hybrid honeypot framework for malware collection and analysis. In *2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5. IEEE, 2012.

[95] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.

[96] Michael J Lee, Bruce Ferwerda, Junghong Choi, Jungpil Hahn, Jae Yun Moon, and Jinwoo Kim. Github developers use rockstars to overcome overflow of news. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 133–138. ACM, 2013.

[97] Roy Ka-Wei Lee and David Lo. Github and stack overflow: Analyzing developer interests across multiple social collaborative platforms. In *International Conference on Social Informatics*, pages 245–256. Springer, 2017.

[98] William Leibzon. Social network of software development at github. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1374–1376. IEEE, 2016.

[99] Corrado Leita, Marc Dacier, and Frederic Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with scriptgen based honeypots. In *International Workshop on Recent Advances in Intrusion Detection*, pages 185–205. Springer, 2006.

[100] Corrado Leita, VH Pham, Olivier Thonnard, E Ramirez-Silva, Fabian Pouget, Engin Kirda, and Marc Dacier. The leurre. com project: collecting internet threats information using a worldwide distributed honeynet. In *2008 WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, pages 40–57. IEEE, 2008.

[101] Toomas Lepik, Kaie Maennel, Margus Ernits, and Olaf Maennel. Art and automation of teaching malware reverse engineering. In *International Conference on Learning and Collaboration Technologies*, pages 461–472. Springer, 2018.

[102] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11(2), 2010.

[103] Longzhuang Li, Yi Shang, and Wei Zhang. Improvement of hits-based algorithms on web documents. In *Proceedings of the 11th international conference on World Wide Web*, pages 527–535, 2002.

[104] Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[105] Zhifang Liao, Haozhi Jin, Yifan Li, Benhong Zhao, Jinsong Wu, and Shengzong Liu. Devrank: Mining influential developers in github. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[106] Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: Github as a collaborative social network. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[107] Yuxuan Liu, Guanghui Yan, Jianyun Ye, and Zongren Li. Community evolution based on tensor decomposition. In *ICPCSEE*, pages 62–75. Springer, 2019.

[108] Tuong Luu. Approach to evaluating clustering using classification labelled data. Master's thesis, University of Waterloo, 2011.

[109] M Madheswaran et al. An improved frequency based agglomerative clustering algorithm for detecting distinct clusters on two dimensional dataset. *Journal of Engineering and Technology Research*, 9(4):30–41, 2017.

[110] GS Mahalakshmi, G MuthuSelvi, and S Sendhilkumar. Gibbs sampled hierarchical dirichlet mixture model based approach for clustering scientific articles. In *Smart Computing Paradigms: New Progresses and Challenges*, pages 169–177. Springer, 2020.

[111] E. Marin, J. Shakarian, and P. Shakarian. Mining key-hackers on darkweb forums. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 73–80, April 2018.

[112] Ericsson Marin, Jana Shakarian, and Paulo Shakarian. Mining key-hackers on darkweb forums. In *ICDIS*, pages 73–80. IEEE, 2018.

[113] Michael Frederick McTear, Zoraida Callejas, and David Griol. *The conversational interface*, volume 6. Springer, 2016.

[114] Sneha Mehta, Mohammad Raihanul Islam, Huzefa Rangwala, and Naren Ramakrishnan. Event detection using hierarchical multi-aspect attention. In *WWW*, pages 3079–3085, 2019.

[115] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[116] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[117] Misterch0c. Misterch0c profile and repositories. https://github.com/misterch0c, 2018.

[118] Misterch0c. Apt34 / oilrig leak, quick analysis. https://misterch0c.blogspot.com/2019/04/apt34-oilrig-leak.html, 2019.

[119] Mitre. State sponsored hacking tool. https://attack.mitre.org/software/S0363, 2019.

[120] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[121] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[122] n1nj4sec. Pupy tool. https://github.com/n1nj4sec/pupy/wiki/, 2015. [Online; accessed 08-February-2020].

[123] Abdelmonim Naway and Yuancheng Li. Android malware detection using autoencoder. *arXiv preprint arXiv:1901.07315*, 2019.

[124] Hoang Nguyen, Xuan-Nam Bui, Quang-Hieu Tran, and Ngoc-Luan Mai. A new soft computing model for estimating and controlling blast-produced ground vibration based on hierarchical k-means clustering and cubist algorithms. *Applied Soft Computing*, 77:376–386, 2019.

[125] Nicolas Verdier. Security researcher. https://www.linkedin.com/in/nicolas-verdier-b23950b6/. [Online; accessed 14-February-2020].

[126] Feiping Nie, Wei Zhu, and Xuelong Li. Unsupervised large graph embedding based on balanced and hierarchical k-means. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[127] Nikhil Gupta. Should we create a separate git repository of each project or should we keep multiple projects in a single git repo? https://www.quora.com/Should-we-create-a-separate-git-repository-of-each-project-or-should-we-keep-multiple-projects-in-a-single-git-repo/answer/Nikhil-Gupta-55, 2019. [Online; accessed 14-February-2020].

[128] OffCom. Offensive community website. http://offensivecommunity.net/, March, 2020 2020. [accessed July-2020].

[129] Online Forums. Ethical hacker, hack this site, offensive community, wilders security, mpgh. https://www.ethicalhacker.net/, https://www.hackthissite.org/, http://offensivecommunity.net/, https://www.wilderssecurity.com/, http://mpgh.net/.

[130] Evangelos Papalexakis and A Seza Doğruöz. Understanding multilingual social networks in online immigrant communities. In *WWW*, page 865, 2015.

[131] Evangelos E Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *SDM16*, pages 711–719. SIAM, 2016.

[132] Sergio Pastrana and Guillermo Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. *arXiv preprint arXiv:1901.00846*, 2019.

[133] Sergio Pastrana, Daniel R Thomas, Alice Hutchings, and Richard Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *WWW*, pages 1845–1854, 2018.

[134] Paul Roberts. Spyeye and zeus malware: Married or living separately? https://threatpost.com/spyeye-and-zeus-malware-married-or-living-separately-101411/75755/, 2011. [Online; accessed 14-March-2020].

[135] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351. ACM, 2014.

[136] Daniel Plohmann, Martin Clauss, Steffen Enders, and Elmar Padilla. Malpedia: a collaborative effort to inventorize the malware landscape. *Proceedings of the Botconf*, 2017.

[137] Rebecca S Portnoff, Sadia Afroz, Greg Durrett, Jonathan K Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. Tools for automated analysis of cybercriminal markets. In *WWW*, page 657, 2017.

[138] Anuja Priyam, GR Abhijeeta, Anju Rathee, and Saurabh Srivastava. Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, 3(2):334–337, 2013.

[139] PyGithub. A python libraray to use github api v3. https://github.com/PyGithub/PyGithub/. [Online; accessed 13-October-2019].

[140] Austen Rainer and Stephen Gale. Evaluating the quality and quantity of data on open source software projects. In *Procs 1st int conf on open source software*, 2005.

[141] Raj Chandel. Article on pupy. https://www.hackingarticles.in/command-control-tool-pupy/, 2019. [Online; accessed 08-February-2020].

[142] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[143] Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 659–661, 2002.

[144] Md Omar Faruk Rokon, Risul Islam, Ahmad Darki, Evangelos E. Papalexakis, and Michalis Faloutsos. Sourcefinder: Finding malware source-code from publicly available repositories in github. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 149–163. USENIX, 2020.

[145] Md Omar Faruk Rokon, Risul Islam, Ahmad Darki, Evangelos E. Papalexakis, and Michalis Faloutsos. Sourcefinder: Finding malware source-code from publicly available repositories in github. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 149–163. USENIX, 2020.

[146] Md Omar Faruk Rokon, Pei Yan, Risul Islam, and Michalis Faloutsos. Repo2vec: A comprehensive embedding approach for determining repository similarity. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021.

[147] Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. *The Journal of Machine Learning Research*, 18(1):3077–3111, 2017.

[148] Hassen Saıdi, Phillip Porras, and Vinod Yegneswaran. Experiences in malware binary deobfuscation. *Virus Bulletin*, 2010.

[149] A. Sapienza, A. Bessi, S. Damodaran, P. Shakarian, K. Lerman, and E. Ferrara. Early warnings of cyber threats in online discussions. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 667–674, Nov 2017.

[150] Anna Sapienza, Alessandro Bessi, and Emilio Ferrara. Non-negative tensor factorization for human behavioral pattern mining in online games. *Information*, 9(3):66, 2018. Multidisciplinary Digital Publishing Institute.

[151] Anna Sapienza, Sindhu Kiranmai Ernala, Alessandro Bessi, Kristina Lerman, and Emilio Ferrara. Discover: Mining online chatter for emerging cyber threats. In *Companion Proceedings of the The Web Conference 2018*.

[152] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 298–307, 2015.

[153] Eric Schulte, Jason Ruchti, Matt Noonan, David Ciarletta, and Alexey Loginov. Evolving exact decompilation. In *Workshop on Binary Analysis Research (BAR)*, 2018.

[154] Security Forums. Ethical hacker, hack this site, offensive community, wilders security. https://www.ethicalhacker.net/, https://www.hackthissite.org/, http://offensivecommunity.net/, https://www.wilderssecurity.com/.

[155] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1055–1064, 2015.

[156] Madhu K Shankarapani, Subbu Ramamoorthy, Ram S Movva, and Srinivas Mukkamala. Malware detection using assembly and api call sequences. *Journal in computer virology*, 7(2):107–119, 2011.

[157] Deepak Sharma, Bijendra Kumar, and Satish Chand. A survey on journey of topic modeling techniques from svd to deep learning. *IJMECS*, 9(7):50, 2017. Modern Education and Computer Science Press.

[158] Victor RL Shen, Chin-Shan Wei, and Tony Tong-Ying Juang. Javascript malware detection using a high-level fuzzy petri net. In *2018 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 511–514. IEEE, 2018.

[159] Kevin Sheridan, Tejas G Puranik, Eugene Mangortey, Olivia J Pinon-Fischer, Michelle Kirby, and Dimitri N Mavris. An application of dbscan clustering for flight anomaly detection during the approach phase. In *AIAA Scitech 2020 Forum*, page 1851, 2020.

[160] Lei-Lei Shi, Lu Liu, Yan Wu, Liang Jiang, Muhammad Kazim, Haider Ali, and John Panneerselvam. Human-centric cyber social computing model for hot-event detection and propagation. *IEEE Transactions on CSS*, 6(5):1042–1050, 2019. IEEE.

[161] Simone Catania. News of simplelocker outbreak. https://www.internetx.com/en/news-detailview/die-10-gefaehrlichsten-ransomware-varianten-der-letzten-jahre/. [Online; accessed 08-February-2020].

[162] Sophron. Wifiphisher. https://github.com/wifiphisher/wifiphisher, 2014. [Online; accessed 14-March-2020].

[163] Will Wei Sun and Lexin Li. Dynamic tensor clustering. *Journal of the American Statistical Association*, 114(528):1894–1907, 2019.

[164] Xiaoyan Sun, Yang Wang, Jie Ren, Yuefei Zhu, and Shengli Liu. Collecting internet malware based on client-side honeypot. In *2008 The 9th International Conference for Young Computer Scientists*, pages 1493–1498. IEEE, 2008.

[165] Dean Teffer, Ravi Srinivasan, and Joydeep Ghosh. Adahash: hashing-based scalable, adaptive hierarchical clustering of streaming data on mapreduce frameworks. *International Journal of Data Science and Analytics*, 8(3):257–267, 2019.

[166] Sachin Thukral, Hardik Meisheri, Tushar Kataria, Aman Agarwal, Ishan Verma, Arnab Chatterjee, and Lipika Dey. Analyzing behavioral trends in community driven discussion platforms like reddit. In *ASONAM*, pages 662–669. IEEE, 2018.

[167] Ferdian Thung, Tegawende F Bissyande, David Lo, and Lingxiao Jiang. Network structure of social coding in github. In *2013 17th European conference on software maintenance and reengineering*, pages 323–326. IEEE, 2013.

[168] Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 809–825. Springer, 2017.

[169] SL Ting, WH Ip, and Albert HC Tsang. Is naive bayes a good classifier for document classification. *International Journal of Software Engineering and Its Applications*, 5(3):37–46, 2011.

[170] Gülen Toker and O Kirmemis. Text categorization using k nearest neighbor classification. *Survey Paper, Middle East Technical University*, 2013.

[171] Tom K. Hacking news of fahim magsi. https://www.namepros.com/threads/hacked-by-muslim-hackers.950924/, 2016. [Online; accessed 08-February-2020].

[172] Tommy Hodgins. Choosing between "one project per repository" vs "multiple projects per repository" architecture. https://hashnode.com/, 2017. [Online; accessed 14-February-2020].

[173] Virus Total. Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en*, 2019.

[174] Christoph Treude, Larissa Leite, and Maurício Aniche. Unusual events in github repositories. *Journal of Systems and Software*, 142:237–247, 2018.

[175] VentureBeat. Github passes 100 million repositories. https://venturebeat.com/2018/11/08/github-passes-100-million-repositories/. [Online; accessed 13-October-2019].

[176] VirusBay. A web-based, collaboration platform for malware researcher. *Online: https://beta.virusbay.io/*, 2019.

[177] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[178] Chi Wang, Xueqing Liu, Yanglei Song, and Jiawei Han. Towards interactive construction of topical hierarchy: A recursive tensor decomposition approach. In *ACM SIGKDD*, pages 1225–1234, 2015.

[179] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[180] Dawid Weiss. Quantitative analysis of open source projects on sourceforge. In *Proceedings of the First International Conference on Open Source Systems, Genova*, pages 140–147. Citeseer, 2005.

[181] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270, 2010.

[182] Wikipedia. Linux based botnet bashlite. https://en.wikipedia.org/wiki/BASHLITE/. [Online; accessed 08-February-2020].

[183] Wikipedia. Shadow broker. https://en.wikipedia.org/wiki/The$_S hadow_B rokers$, 2020.

[184] WS. Wilders security website. https://www.wilderssecurity.com/, March, 2020 2020. [accessed July-2020].

[185] Joicymara Xavier, Autran Macedo, and Marcelo de Almeida Maia. Understanding the popularity of reporters and assignees in the github. In *SEKE*, 2014.

[186] Hangjun Xu. An algorithm for comparing similarity between two trees. *arXiv preprint arXiv:1508.03381*, 2015.

[187] Ji Xu, Guoyin Wang, and Weihui Deng. Denpehc: Density peak based efficient hierarchical clustering. *Information Sciences*, 373:200–218, 2016.

[188] Shuo Xu. Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.

[189] Y. Nativ and S. Shalev. Github repository: thezoo. https://github. com/ytisf/theZoo, 2014. [Online; accessed 14-March-2020].

[190] Khaled Yakdan, Sergej Dechand, Elmar Gerhards-Padilla, and Matthew Smith. Helping johnny to analyze malware: A usability-optimized decompiler and malware analysis user study. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 158–177. IEEE, 2016.

[191] Yuval Nativ. Security researcher. https://morirt.com/. [Online; accessed 14-February-2020].

[192] Lenny Zeltser. Free malware sample sources for researchers. https://zeltser.com/malware-sample-sources/. [Online; accessed 13-October-2019].

[193] Jixin Zhang, Zheng Qin, Hui Yin, Lu Ou, and Yupeng Hu. Irmd: malware variant detection using opcode image recognition. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1175–1180. IEEE, 2016.

[194] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[195] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[196] Yong-liang Zhao and Quan Qian. Android malware identification through visual exploration of disassembly files. *International Journal of Network Security*, 20(6):1061–1073, 2018.

[197] Xingsi Zhong, Yu Fu, Lu Yu, Richard Brooks, and G Kumar Venayagamoorthy. Stealthy malware traffic-not as innocent as it looks. In *2015 10th International Conference on Malicious and Unwanted Software*, pages 110–116. IEEE, 2015.

[198] Jianwei Zhuge, Thorsten Holz, Xinhui Han, Chengyu Song, and Wei Zou. Collecting autonomous spreading malware using high-interaction honeypots. In *International Conference on Information and Communications Security*, pages 438–451. Springer, 2007.