Proactively Identifying Emerging Hacker Threats from the Dark Web: A Diachronic Graph Embedding Framework (D-GEF)

SAGAR SAMTANI, Department of Operations and Decision Technologies, Indiana University
HONGYI ZHU, Department of Information Systems and Cyber Security, University of Texas at San Antonio
HSINCHUN CHEN, Department of Management Information Systems, University of Arizona

Cybersecurity experts have appraised the total global cost of malicious hacking activities to be $450 billion annually. Cyber Threat Intelligence (CTI) has emerged as a viable approach to combat this societal issue. However, existing processes are criticized as inherently reactive to known threats. To combat these concerns, CTI experts have suggested proactively examining emerging threats in the vast, international online hacker community. In this study, we aim to develop proactive CTI capabilities by exploring online hacker forums to identify emerging threats in terms of popularity and tool functionality. To achieve these goals, we create a novel Diachronic Graph Embedding Framework (D-GEF). D-GEF operates on a Graph-of-Words (GoW) representation of hacker forum text to generate word embeddings in an unsupervised manner. Semantic displacement measures adopted from diachronic linguistics literature identify how terminology evolves. A series of benchmark experiments illustrate D-GEF's ability to generate higher quality than state-of-the-art word embedding models (e.g., word2vec) in tasks pertaining to semantic analogy, clustering, and threat classification. D-GEF's practical utility is illustrated with in-depth case studies on web application and denial of service threats targeting PHP and Windows technologies, respectively. We also discuss the implications of the proposed framework for strategic, operational, and tactical CTI scenarios. All datasets and code are publicly released to facilitate scientific reproducibility and extensions of this work.

CCS Concepts: • **Security and privacy → Malware and its mitigation**;

Additional Key Words and Phrases: Cyber threat intelligence, hacker forums, ransomware, deep learning, text graphs, graph convolutions, diachronic linguistics, graph embeddings

This material is based upon work supported by the National Science Foundation under Grant No. DUE-1303362 (SFS), SES-1314631 (SaTC), CNS-1850362 (CRII SaTC), and CNS-1917117 (CICI).

Author's addresses: S. Samtani (corresponding author), Department of Operations and Decision Technologies, Indiana University, 1275 E. 10th St., Bloomington, Indiana 47405; email: ssamtani@iu.edu; H. Zhu, Department of Information Systems and Cyber Security, University of Texas at San Antonio, 1 UTSA Circle, San Antonio, TX 78249; email: hongyi.zhu@utsa.edu; H. Chen, Department of Management Information Systems, University of Arizona, 1130 E. Helen St., McClelland Hall 430, Tucson, AZ 85721; email: hsinchun@email.arizona.edu.

2471-2566/2020/08-ART21 $15.00
https://doi.org/10.1145/3409289

ACM Reference format:
Sagar Samtani, Hongyi Zhu, and Hsinchun Chen. 2020. Proactively Identifying Emerging Hacker Threats from the Dark Web: A Diachronic Graph Embedding Framework (D-GEF). *ACM Trans. Priv. Secur.* 23, 4, Article 21 (August 2020), 33 pages.
https://doi.org/10.1145/3409289

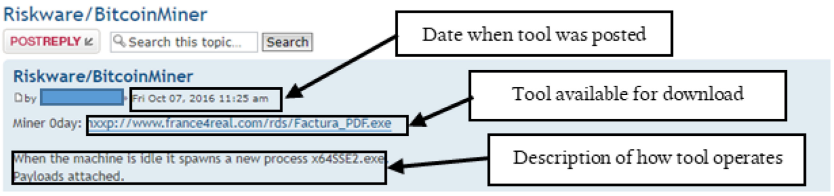
1 INTRODUCTION

Computing technology has afforded modern society with numerous benefits. Many private and public organizations employ complex information systems (IS) to execute financial transactions, maintain health records, and control critical infrastructure. Unfortunately, the rapid integration of IS has been met with an alarming rate of cyber-attacks conducted by malicious hackers. Cybersecurity experts have appraised the total annual cost of hacktivism, espionage, cyberwarfare, and other cybercrime hacking activities against major entities such as Equifax, Uber, and Yahoo! at $450 billion. To combat this significant societal issue, many organizations have started developing and using Cyber Threat Intelligence (CTI).

CTI is a data-driven process that focuses on developing timely, relevant, and actionable intelligence about emerging threats (e.g., exploits, vulnerabilities, etc.) and key threat actors (i.e., hackers) to enable effective cybersecurity decisions [11]. Prevailing CTI procedures collect data from Network Intrusion Detection/Prevention Systems (NIDS/NIPS), and log files generated from servers, workstations, firewalls, databases, and other internal network devices. Established analytics such as event correlation, forensics, anomaly detection, malware analysis, and others are applied to collected data to generate intelligence about the threats used against the networks. Despite the maturity and value of these processes, the data analyzed are past network events. Consequently, the derived intelligence is inherently reactive to known threats. These limitations have motivated CTI professionals from the acclaimed SANS Institute to note that "most organizations are still reactive to alerts and incidents instead of proactively seeking out the threats" [26]. Consequently, the quantity, severity, and sophistication of threats used in cyber-attacks increase annually.

To combat these concerns, CTI experts have suggested proactively examining emerging threats in the vast, international online hacker community [9, 49]. The online hacker community motivates millions of hackers from the US, China, Russia, and the Middle East to share malicious tools and knowledge. Today, four major hacker community platforms exist: forums, DarkNet Marketplaces, Internet-Relay-Chat (IRC) channels, and carding shops [11]. Although each has CTI value, hacker forums are particularly useful to CTI experts. Unlike other platforms, forums allow hackers to freely share and discuss cyber-attack threats. Figure 1 illustrates one example, where a hacker provides Bitcoin miner 0-day threats for other hackers to freely download and use.

Hackers have used threat knowledge and content available in hacker forums to execute well-known breaches. One notable example is the Target incident, where hackers procured the Black-POS malware from forums months before executing the attack. This event's severity helps motivate the careful analysis of hacker forum data to identify emerging threats. However, hacker forums contain hundreds of thousands of unstructured, un-sanitized text records. Hackers rapidly evolve in their skillsets; thus, they develop new malware and augment existing threats with novel functions. Compounding this issue are the unclear semantics of hacker terminology (e.g., "injection" can refer to memory, SQL, or process), and how they shift over time. Prevailing CTI analytics such as IP reputation services and event correlation are ill-equipped for these unique characteristics. Moreover, conventional and emerging text analytics approaches employed in extant hacker forum literature require significant extensions to generate valuable CTI. These challenges present


POSTREPLY
Search this topic...
Search
Date when tool was posted
Tool available for download
Description of how tool operates

Riskware/BitcoinMiner

Riskware/BitcoinMiner

by [redacted] » Fri Oct 07, 2016 11:25 am

Miner 0day: hxxp://www.france4real.com/rds/Factura_PDF.exe

When the machine is idle it spawns a new process x64SSE2.exe. Payloads attached.

Fig. 1. Hacker Providing a Bitcoin Miner 0-day **Threat** on a Hacker Forum.

numerous challenges for CTI professionals and motivate the development of innovative CTI text analytics.

In this study, we aim to develop critically needed proactive CTI capabilities by exploring online hacker forums to identify emerging threats in terms of popularity and functionality. To achieve these objectives, we draw upon emerging text mining, graph embedding, network science, and diachronic linguistics methods to design, develop, evaluate, and demonstrate a novel Diachronic Graph Embedding Framework (D-GEF). As a result of these processes, this article makes the following contributions:

- First, the proposed D-GEF operates on a novel directed Graph-of-Words (GoW) representation of hacker forum text.
- Second, we operate state-of-the-art unsupervised graph embedding algorithms upon the hacker forum GoW to automatically generate low-dimensional word embeddings.
- Third, and relatedly, we perform a series of rigorous evaluations to identify how graph embeddings can generate higher quality embeddings than state-of-the-art Neural Network Language Models (NNLMs) such as word2vec in tasks pertaining to semantic relatedness, semantic analogy, clustering, and threat classification.
- Fourth, semantic displacement measures are adopted from diachronic linguistics literature identifying how word embeddings evolve over time to pinpoint emerging threats in terms of popularity and functionality.
- Fifth, we illustrate D-GEF's practical utility with an in-depth case study identifying emerging trends and functionalities of web application and denial of service (DoS) threats targeting PHP and Windows technologies (respectively) in a large-scale, international hacker forum.
- Finally, we publicly release all datasets and code to facilitate scientific reproducibility and extensions of this work.

This article is organized as follows: First, we review literature related to the online hacker community to identify key forum features and past efforts in detecting emerging threat trends. Second, we summarize key gaps from extant literature and pose several research questions for study. Third, we review text graphs, graph embeddings, and diachronic word embeddings to ground and guide our proposed D-GEF framework. Fourth, we present each component of the D-GEF. Fifth, we summarize our evaluation procedure, including experiment designs and benchmark datasets. Subsequently, we summarize our evaluation and illustrate the potential utility with in-depth case studies, discuss their security implications, and offer suggestions on promising directions for future research. The last section concludes this work.

2 RELATED WORK: HACKER COMMUNITY RESEARCH

As mentioned in the introduction, hackers often use DarkNet Marketplaces, forums, carding shops, and IRC channels to share malicious tools and knowledge [8, 11, 43]. Among these, hacker forums

Table 1. Selected Studies Identifying Threats in Online Hacker Forums

Year	Author	Data Source	Analytics*	Selected Identified Threats	Threat Trend Identification?	Trend Identification Method
2019	Pastrana and Saurez-Tangil [34]	Forums, VirusTotal	Keyword based	Cryptomining malware	Yes	Term frequency
2018	Yuan et al. [60]	Four forums	Word2vec	RAT, Zeus	No	None
2018	Goyal et al. [15]	Forums, Twitter, Blogs	RNN, LSTM	Phishing, Windows exploits, trojans	Yes	Term forecasting
2018	Williams et al. [57]	10 forums	LSTM	Web, database, mobile, network, system	Yes	Term frequency
2018	Zhang et al. [62]	One forum	HIN	Phishing, SQLi, crypters, web exploits	No	None
2017	Durrett et al. [12]	Three forums	Domain adaptation	Bots, spam, kits	No	None
2017	Yang et al. [58]	Forums, markets	Keyword approach	Domains, fraudulent content	No	None
2017	Sapienza et al. [46]	Twitter, forums	Keyword approach	Botnets, DDoS	Yes	Term frequency
2017	Samtani et al. [43]	Eight forums	LDA and SVM	Keyloggers, DDoS, SQLi, web exploits	Yes	Term frequency
2017	Grisham et al. [16]	Four forums	RNN	Mobile malware	Yes	Term frequency
2016	Li et al. [27]	Three forums	sLDA	Phishing, botnets	No	None
2016	Nunes et al. [31]	21 forums	SVM	Botnets, keyloggers, worms, 0-days	No	None
2016	Zhao et al. [63]	29 QQ Groups	Word2vec, LDA	Carding materials	No	None
2016	Zhao et al. [64]	12 forums	Manual	DDoS, web exploits	No	None
2016	Samtani et al. [41]	Two forums	LDA and SVM	Bots, crypters, keyloggers, bank exploits	Yes	Term frequency
2015	Hutchings and Holt [23]	13 forums	Manual	Keyloggers, banking Trojans	No	None

**Note*: DDoS=Distributed Denial of Service; HIN=Heterogenous Information Network; LDA=Latent Dirichlet Allocation; LSTM=Long Short-Term Memory; RAT=Remote Administration Tool; RNN=Recurrent Neural Network; sLDA= supervised Latent Dirichlet Allocation; Structured Query Language Injection=SQLi; SVM=Support Vector Machine.

are particularly valuable for detecting emerging threats. Carding shops and IRC platforms do not provide the mechanisms for hackers to freely share threats, while DarkNet Marketplaces have more drug, pornography, and weapon material than cybersecurity-related content [11]. Forums also provide richer metadata, namely postdate and post content. Such features are not consistently available on other platforms. These characteristics have motivated numerous researchers to identify hacker forums to proactively identify threats. Table 1 summarizes selected recent studies based on the datasets used, analytics run, identified threats, and if and how trends were identified. For purposes of the scope and goals of this research, we limit our review to selected studies operating on hacker forums.

Nearly all studies have been exploratory in nature, with the goal of identifying what content is available. Scholars have employed support vector machine (SVM), topic modeling, heterogeneous information networks (HIN), keyword approaches, and interviews with subject matter experts (SMEs) to identify threats in forums [23, 41, 42, 64]. Analyses reveal that hackers freely share threats such as botnets, email hacks, kits, keyloggers, remote administration tools (RATs), bank threats, denial of service (DoS), and many others. Several studies have gone one step beyond identifying threats to detecting the overall trends. For example, Grisham et al. [16] plotted the number of mobile malware occurrences within a major Arabic hacker forum. Other studies have monitored the frequency of threat terms (e.g., "botnet," "crypter," etc.) over a selected time period [41, 43, 46].

While providing CTI value, using term frequency, bag-of-words, and keyword approaches have several limitations. First, a term's context is ignored. For example, "injection" can refer to "SQL" or to "memory." Consequently, results can lack granularity. Second, hackers' rapid expansion of their vocabulary can result in these methods initially overlooking new threat terms. One example is Mirai, which appeared months before the attack. However, it remained undetected as cybersecurity professionals were unaware of the new botnet term. Finally, term frequencies and bag-of-words representations cannot capture the distance, context, or relationships of terms within and across forum posts [36]. While word2vec offers a mechanism to project terms as vector representations (i.e., embeddings) into a low-dimensional space and potentially help address some of these issues, past studies have only used it in a synchronic fashion, wherein embeddings are only analyzed in one time point. As a result, it is unclear how embeddings shift (i.e., semantics change) and relationships evolve across time periods. Taken together, these limitations prevent an understanding of the depth, breadth, complexity, and evolution of hacker language. Therefore, we pose the following research questions for study.

- How can online hacker forum text be represented in a way that captures the relationships of terms within and across all of a forum's posts?
- How can this representation be used to identify the emerging trends in online hacker forums?
- How do the semantics of hacker terminology shift over time?

3 METHODOLOGICAL FOUNDATION FOR PROPOSED D-GEF

The limitations summarized above necessitate an alternative approach to represent, process, and longitudinally analyze hacker forum text. In the following sub-sections, we summarize three methodological components that form the basis of our proposed D-GEF. First, we review text graphs to identify approaches for and benefits of representing text as a network. Second, we examine prevailing graph embedding methods as a mechanism to automatically extract embeddings from text graphs to facilitate semantic analysis. We focus the review on unsupervised methods only as they are the most ideal for dynamically generating hacker forum text embeddings across multiple time-spells of data without relying on external gold-standard datasets for training. Finally, diachronic word embeddings are reviewed to identify approaches on how to compare embeddings across multiple time-spells to detect semantic evolutions of hacker terminology.

3.1 Text Graphs: Graph of Words (GoW) Representation

Text graphs are gaining significant traction within the Natural Language Processing (NLP) community due to their ability to capture and reveal relationships, patterns, and regularities within a corpus not captured in standard representations (e.g., bag-of-words) and language models (e.g., term frequency-inverse document frequency). Text graphs build upon network science principles to represent text in a graph. Like other network science applications, text graphs are constructed with nodes and edges. Nodes are text units such as words, collocations, word senses, sentences, or documents. Edges are the relationships between text units. One prevalent text graph formulation is the word co-occurrence network, also known as the Graph-of-Words (GoW) [30]. Nodes in this network are words, and edges indicate whether two words appear in a specified text unit (e.g., document, sentence, etc.). Weighted edges denote how frequently two words have co-occurred.

All GoW node relationships are held in an NxN adjacency matrix A, where N denotes the number of nodes (i.e., words) [5]. If a relationship exists between words i and j (i.e., words i and j appear in the same text unit), A_{ij} is 1. If no relationship is present, A_{ij} is 0. This representation enables researchers to calculate a suite of network and node-level descriptive statistics. Both provide insight into the richness, expressiveness, and universality of a corpus's vocabulary. Network-level metrics (e.g., network density, clustering coefficient, etc.) provide insight into the diversity, breadth, and depth of a corpus's vocabulary. Node-level measures (e.g., degree, eigenvector, betweenness, and closeness) pinpoint key words based on different criteria. Count-based metrics (degree and eigenvector) sum and/or weight the number of in- and out-links from a node. Betweenness and closeness metrics measure the distances of nodes.

Scholars have integrated selected measures as features to achieve state-of-the-art performances in authorship analysis [2] and text classification [38]. However, recent years have seen efforts shift from manual feature engineering (often ad-hoc, labor- and time-intensive) to automatic generation of latent, low-dimensional vector representations (i.e., embeddings). These embeddings aim to comprise a condensed feature representation for each node or edge on a network. For hacker forums' GoWs, graph embeddings can reveal latent local and global relationships of threat terms that would otherwise be overlooked. Moreover, it can facilitate the longitudinal analysis of how terminology semantically shifts over time. For these reasons, we review prevailing unsupervised graph embedding approaches next.

3.2 Unsupervised Graph Embedding Methods

Graph embedding methods rely on a series of mathematical transformations to project the graph into a low-dimensional space. Most commonly, these methods can operate in a supervised or unsupervised manner. Supervised approaches require a gold-standard dataset from which it can create mappings between the training data and the pre-specified output labels. While the created embeddings are valuable for the specified task, they are less suitable for or generalizable to other objectives (e.g., generating an embedding for a word in a GoW). Given this study's goal, this drawback requires an unsupervised approach that generates task-independent node embeddings without a gold-standard dataset. Such a resource is not available for all corpora, especially emerging cybersecurity-relevant content.

Unsupervised graph embedding approaches aim to create a low-dimensional embedding without using any external resources (e.g., gold-standard training data). Embeddings are created by preserving node proximities at varying order levels. For example, first-order proximity preserves edges, second-order proximity preserves a node's similarity to its direct neighborhood, and higher-order preserves the direct neighborhood of an adjacent node. Proximities can be defined at local and global levels. These proximities are optimized by an objective function to create embeddings. In general, four categories of objective functions exist:

1. **Matrix factorization:** Uses a series of matrix operations (e.g., singular value decomposition) on selected matrices generated from a graph (e.g., adjacency, degree, etc.)
2. **Random walk-based:** Estimates the probability of visiting a node from a specified graph location using a walking strategy.

Table 2. Summary of Prevailing Unsupervised Graph Embedding Algorithms

Category	Model	Projection Method	Edge Variations				Proximity Preserved		References
			D	U	W	UW	Global	Local	
Matrix Factorization	Laplacian Eigenmaps	Embedding space spanned by significant eigenvectors		√	√	√	–	1st	[6]
	Graph Factorization	Embedding inner products approximate edge weights between nodes	√	√	√	√	–	1st	[1]
	LLE	Embedding space spanned by significant eigenvectors	√	√	√	√	–	1st	[22]
	HOPE	Factorize high-order proximity matrix	√	√	√	√	High	1st	[32]
	GraRep	Aggregate a graph's k-step representations obtained from factorization	√	√	√	√	High	1st	[10]
Random Walk-based	DeepWalk	Uniform walks; skip-gram	√	√	√	√	2nd	–	[35]
	Node2vec	Biased walks; skip-gram	√	√	√	√	2nd	–	[17]
Deep Rep. Learning	SDNE	Autoencoder; reconstruct adjacency matrix	√	√	√	√	2nd	1st	[55]
	VGAE	Variational Autoencoder; generative model		√		√		1st	[25]
Edge Recon.	LINE	Explicit 1st- and 2nd-order proximity feature extraction; negative edge sampling	√	√	√	√	2nd	1st	[51]

* *Note*: LLE = Local Linear Embedding; HOPE = High-Order Proximity preserved Embedding; GraRep = Graph Representation; SDNE = Structural Deep Network Embedding; VGAE = Variational Graph Autoencoder; LINE = Large-scale Information Network Embedding; D = Directed; U = Undirected; W = Weighted; UW = Unweighted.

3. **Deep representation learning:** Relies on an unsupervised deep learning approach (e.g., autoencoders) to create embeddings via feed-forward, error correction, and backpropagation.
4. **Edge reconstruction:** Samples edges based on weighted edges and nodes to preserve local and global proximities.

Methods in each category may use multiple approaches to optimize objective functions and create embeddings. Table 2 summarizes the prevailing models in each category that aim to create an embedding for nodes, as it is most closely related to the proposed work. For each method, we summarize its projection method and its ability to operate on directed, undirected, weighted, or unweighted edges. We also provide a brief summary of the global and local properties preserved in the projection process.

Selection of graph embedding depends on graph type. For example, Laplacian eigenmaps and VGAE are not designed for directed graphs. Irrespective of the approach, the embeddings generated are suitable for exploratory and/or descriptive analysis. Moreover, they can be inputted into and support other downstream tasks with well-established operational procedures (e.g., classification and clustering). Within this study, the specified downstream task is mapping how the graph-of-words embeddings shift to identify semantic shifts of hacker terminology. To this end, we review diachronic word embeddings.

3.3 Diachronic Word Embeddings

Despite their promise, extant graph embedding methods cannot capture embedding evolution, shifts, or changes in temporal datasets. While data can be split into time-spells and embeddings created in each, each time-spell would have a different semantic embedding space. These differences prevent the direct comparison of embeddings across time-spells. Thus, embedding spaces need alignment via an external mechanism to enable fair and accurate comparisons. In this study, operating graph embedding methods on a GoW would result in a novel approach to creating word embeddings. Using this perspective, we examine an emerging stream of literature from information retrieval and linguistics aligning embedding spaces over multiple time-spells to model semantic shifts, i.e., diachronic word embeddings.

While synchronic linguistics studies language at one time point, diachronic (i.e., historical) linguistics examines language development and evolution. While traditionally reliant on manual approaches, the advent of the synchronic unsupervised Neural Network Language Model (NNLM) word2vec by Mikolov et al. [29] has spurred a new area of academic inquiry: diachronic word embeddings [18]. As alluded to in the previous sub-section, one can split temporal data into multiple time-spells and create low-dimensional synchronic word embeddings in each. While semantic similarity of word embeddings within time-spells can be compared, the same cannot be done across spells, as the embeddings will not be naturally aligned (i.e., projected into the same semantic spaces).

The prevailing method to align embedding spaces constructs a matrix of word embeddings at each time-spell, $\mathbf{W}^{(t)} \in \mathbb{R}^{d \; x \; |V|}$, where t is the time-spell [18]. Matrices generated at two time-spells are aligned using the solution to the orthogonal Procrustes problem. Specifically, embedding spaces are aligned across time-periods while preserving cosine similarities by optimizing:

$$\mathrm{R}^{(t)} = \underset{Q^T Q = I}{argmin} \; \|\mathbf{W}^{(t)}\mathbf{Q} - \mathbf{W}^{(t+1)}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This solution conforms to the best rotational alignment of both embedding spaces and can be attained by using an application of Singular Value Decomposition (SVD) [48]. Aligning spaces facilitates the computation of a novel linguistic metric to model language evolution: semantic displacement. Semantic displacement measures a word's semantic shift across time-periods by measuring the cosine distance of a word at two time-periods (i.e., cosine-dist(w_t, $w_{t+\Delta}$)). Computing this value across all time-spells shows a word's rate of semantic shift (i.e., how a word evolves in its usage). Taken together, diachronic word embedding approaches have enabled scholars to identify how German, French, and Chinese change across the centuries on the Google books corpus [18], the evolution of English on the Corpus of Historical American English [19], and the evolution of terminology usage in *New York Times* articles [59].

4 PROPOSED DIACHRONIC GRAPH EMBEDDING FRAMEWORK (D-GEF)

GoWs' ability to capture relationships between text units can reveal the richness, diversity, and expansiveness of hacker forum content. Graph embeddings provide a valuable mechanism for projecting hacker forum text into low-dimensional spaces to facilitate selected downstream tasks. However, when operating on temporal data, these embedding spaces must be augmented with the diachronic computations to ensure appropriate comparisons of embeddings and facilitate the identification of semantic shifts.

Recognizing the limitations of prior approaches, we propose a novel D-GEF to address these methodological drawbacks and address this study's research questions. The proposed D-GEF has three major components: (1) Time-Spell and Threat Text Graph Construction, (2) Node

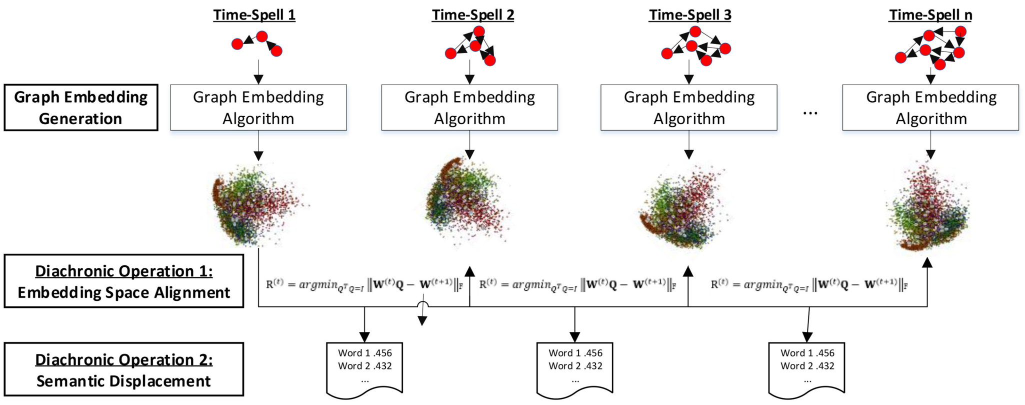

Time-Spell 1
Time-Spell 2
Time-Spell 3
Time-Spell n
Graph Embedding Generation
Graph Embedding Algorithm
Graph Embedding Algorithm
Graph Embedding Algorithm
...
Graph Embedding Algorithm
Diachronic Operation 1: Embedding Space Alignment
$R^{(t)} = argmin_{Q^TQ=I} \|W^{(t)}Q - W^{(t+1)}\|_F$
$R^{(t)} = argmin_{Q^TQ=I} \|W^{(t)}Q - W^{(t+1)}\|_F$
$R^{(t)} = argmin_{Q^TQ=I} \|W^{(t)}Q - W^{(t+1)}\|_F$
Diachronic Operation 2: Semantic Displacement
Word 1 .456
Word 2 .432
...

Fig. 2. Proposed Diachronic Graph Embedding Framework (D-GEF).

Embedding Generation, and (3) Diachronic Operations. Figure 3 depicts the conceptual architecture of the proposed D-GEF. Each component is summarized in the following sub-sections. All D-GEF computations (e.g., text graph generation, embedding, diachronic operations) were implemented using the PyTorch, Numpy, Scikit-learn, Networkx, and Natural Language Toolkit (NLTK) packages in Python. We also provide the full codebase such that interested researchers can build upon the D-GEF for future research.

4.1 Time-Spell and Threat Text Graph Construction

Conducting graph embedding-based diachronic linguistics requires splitting a collected dataset into multiple time-spells. Time-splits can be made based on key end-user requirements. Following the splitting of a dataset into multiple time-spells, a GoW is constructed in each time-spell to facilitate diachronic analysis. We select a GoW representation as it captures relationships between words missed in other text representations (e.g., vector space model) or word embedding approaches. Formally, each GoW is denoted as $G = (V, E)$. G is the entire directed graph. V is the node set, $\{v_1, v_2, v_3, \ldots, v_n\}$ of all words appearing in posts in that time-spell. E is the edge set, $\{e_1, e_2, e_3, \ldots, e_m\}$. Nodes have an edge if they are adjacent. Directionality is determined by the ordering of the words. Edges are assigned if they appear in the same forum post-description.

Like other social media platforms (e.g., Twitter, Reddit), hacker forum text has considerable inconsistencies and noise [43]. We address these issues with a series of automated pre-processing procedures. First, all words are tokenized based on whitespace. Second, all punctuation is stripped from the post-content to remove extraneous, irrelevant characters. This includes removing URLs. Third, a stop-words list filters all generic terms (e.g., the, at, this). These pre-processing steps are consistent with past literature employing word embedding approaches in hacker forum contexts [7, 52].

The GoW in each time-spell builds on the previous period's; therefore, the graphs are always expanding and never decreasing in size. This ensures there is no information loss. Even if a word appears in a time-spell's posts, does not in the second, and reappears in the following, the graph will retain the existence of the word across all spells. Algorithm 1 presents a pseudocode of how the text graphs are constructed.

To demonstrate how the threat text graph procedure is operationalized, Figure 3 presents an illustrative example of how a sample GoW is constructed with the proposed specification. The top of the figure represents one time-spell, while the bottom presents the subsequent spell. The left side of the figure shows selected example sample posts, while the right side presents a visual

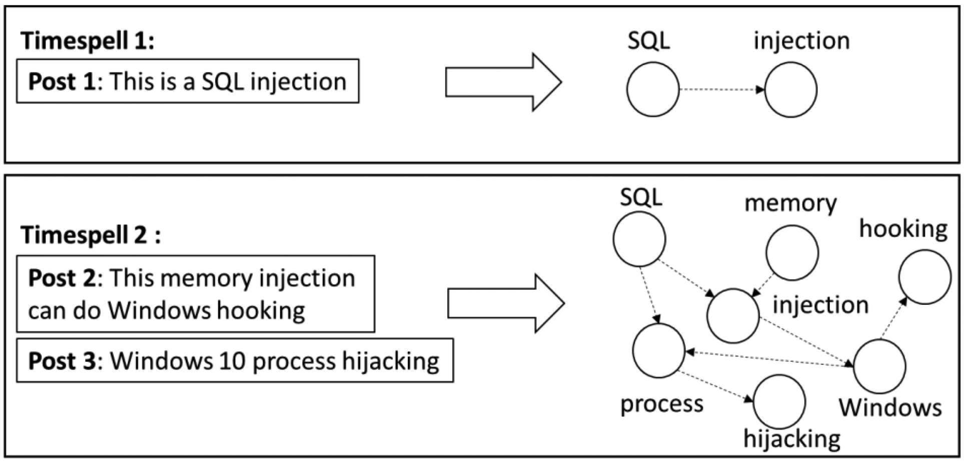

Timespell 1:
Post 1: This is a SQL injection
SQL
injection
Timespell 2 :
Post 2: This memory injection can do Windows hooking
Post 3: Windows 10 process hijacking
SQL
memory
hooking
injection
process
Windows
hijacking

Fig. 3. Example of GoW with the D-GEF specification.

depiction of the generated text graphs. The examples presented are intentionally simplistic; the posts in hacker forums are typically far more extensive and detailed.

ALGORITHM 1: Directed Threat Text Graph Construction

```
Input: Directed text graph of the previous time-spell $G_{t-1} = \{Node, Edge_{t-1}\}$, new documents $D_t = \{doc_1, doc_2, \ldots, doc_i\}$ in time-spell $t$, embedding dimension $d$,
Output: Directed text graph $G_t$ at time-spell $t$
Procedure
    foreach $doc_i$ in $D_t$:
        $doc'_i$ = preprocess($doc_i$) = $(w_i^1, w_i^2, \ldots, w_i^j)$ // Tokenize and remove URL, punctuations, and stop-words from each document
        $Node = Node + w_i^j, \forall w_i^j \notin Node$ // Append new words to the $Node$ set
    end for
    $Edge_t = Edge_{t-1}$
    foreach $doc'_i$:
            $Edge_t = Edge_{t-1} + edge(w_i^k, w_i^{k+1}), \forall\ edge(w_i^k, w_i^{k+1}) \notin Edge_{t-1}$ // Append new directed edge to the updated set $Edge_t$
    end for
    $G_t = \{Node, Edge_{t-1}\}, |Node| = N$ // Construct the new directed text graph $G_t$ for time-spell $t$
    return $G_t$
```

Constructing text graphs in the proposed fashion omits general, unrelated forum discussions while providing a granular look at threat terms and their relationships. It also provides access to an array of metrics to understand overall network dynamics (e.g., vocabulary size, richness, etc.) and pinpoint key network nodes (e.g., key functions, etc.). Table 3 presents selected nodal and topological metrics that can be computed based on the proposed formulation. For each metric, we also provide a brief description of its security-related implications as it pertains to detecting emerging threats from hacker forums.

Measuring these metrics over time can provide insight on how a threat vocabulary evolves, grows, or dissipates over time. Each has significant security implications, including providing critical intelligence on when and how to deploy appropriate security controls to proactively mitigate appropriate security controls. In this vein, the GoW can also be visualized (e.g., Figure 3) to provide systems administrators, Security Operations Center (SOC) analysts, and other CTI professionals

Table 3. Summary of Selected GoW Metrics and Their Security Implications for Diachronic Hacker Forum Threat Analysis

Category	Metric	Definition	Security Implications
Node	Number	# of nodes in the network	Quantity of threat terms
	In-Degree	# of nodes pointing to a given node	Importance of a threat term in the vocabulary (e.g., key functions)
	Out-Degree	# of nodes a given node is pointing to	A threat term's dependence on other terms (e.g., function dependencies)
	Overall Degree	Sum of a node's in- and out- degree	Overall importance of each threat term within the vocabulary
	Betweenness	# of shortest paths passing through a node divided by all shortest paths	Role the threat term plays in holding a vocabulary together
	Eigenvector	Summed connections to others weighted by centrality	How well connected a threat term is across the vocabulary
	Closeness	Avg. # of hops required to reach every other node on the network	Measures how quickly a threat term can reach others
Network	Graph Density	Sum of edges divided by number of possible edges	Relationship and inter-dependence of threat terms
	Eccentricity	Maximum geodesic distance from a node to all other nodes	Similar to closeness; measures how quickly a threat term can reach another
	Diameter	Maximum eccentricity	Indicates the breadth and diversity of a threat vocabulary
	Radius	Minimum eccentricity	
	Avg. path length	Average distance between two nodes	Identifies the average dependencies between threat terms
	Clustering Coefficient	Identifies how nodes tend to cluster together	Pinpoints how threat terms cluster together → indicates the diversity of the threat landscape

the ability to make agile decisions. The operation of the D-GEF framework on varying security datasets may result in varying security implications.

4.2 Graph Embedding Generation

Constructing text graphs and computing network and node-level metrics at each time-spell enables an unprecedented ability to identify emerging threat trends. However, it cannot identify semantic shifts of terms (i.e., new meanings for existing words). To this end, the second phase of the proposed D-GEF aims to automatically generate embeddings for each node in the proposed GoW. Given the general-purpose nature of the proposed framework, we do not specify the exact embedding algorithm to be used for this task. However, two key design considerations must be accounted for. First, since there is no gold-standard dataset, the selected algorithm must operate in an unsupervised fashion. This also ensures that task-independent (i.e., general) node embeddings for GoWs are created. Second, the selection of the algorithm must account for directed graphs. Algorithm 2 presents pseudocode for creating node embeddings with these considerations.

Candidate algorithm selections that adhere to the required design specifications include random walk-based DeepWalk and Node2vec, matrix factorization approaches such as LLE and HOPE, deep-representation learning techniques such as SDNE, and edge reconstruction methods such as LINE. Irrespective of algorithm selection, once node embeddings are generated and tabulated into matrix form in each time-spell (i.e., $\mathbf{W}^{(t)} \in \mathbb{R}^{d\ x\ |V|}$, where t is the time-spell). This compilation supports the subsequent proposed diachronic operations.

4.3 Diachronic Operations: Embedding Alignment and Semantic Shifts

After the embeddings are generated at each time-spell, the diachronic component of the D-GEF performs two tasks: align embedding spaces and compute semantic shifts. For the former, we adopt

ALGORITHM 2: Node Embedding Generation

```
Input: Directed text graph $G_t = \{Node, Edge_t\}$ at time-spell $t$, embedding dimension $d$
Output: Node embedding matrix $E_t = [e_1^t, e_2^t, \dots, e_N^t]$ at time-spell $t$
Procedure
    Define Similarity measure $S_{ij} = f_0(\text{Feature}(Node_i),\ \text{Feature}(Node_j)) \circ f_1\left(edge(Node_i, Node_j)\right) \circ f_2\left(\text{First-order\_Neighbor}(Node_i, Node_j)\right) \circ \dots \circ f_k\left(\text{(k-1)th-order\_Neighbor}(Node_i, Node_j)\right)$ // Define node similarity on graph
    $E_t = [e_1^t, e_2^t, \dots, e_N^t], e_i^t \in \mathbb{R}^d$ // Construct $d$ dimensional node embedding matrix for $G_t$
    $\min_{E_t} \sum_i \sum_j \sum_k \left|dist(e_i^t, e_j^t) - S_{ij}\right|^k$ // Approximate the similarity measure in $G_t$
    return $E_t$
```

orthogonal Procrustes matrix operations as outlined by [18]. Specifically, embedding spaces across time-spells are aligned while retaining cosine similarities by optimizing the following objective function:

$$\mathrm{R}^{(t)} = \underset{Q^TQ=I}{argmin}\ \|\mathbf{W}^{(t)}\mathbf{Q} -\ \mathbf{W}^{(t+1)}\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Algorithm 3 summarizes the pseudocode for aligning embedding spaces.

ALGORITHM 3: Diachronic Embedding Alignment

```
Input: Directed text graph of the previous time-spell $G_{t-1}$, embedding matrix of the previous time-spell $E_{t-1} = [e_1^{t-1}, e_2^{t-1}, \dots, e_M^{t-1}]$, where $e_i^{'}$ is the embedding for node $Node_i$, embedding matrix of the current time-spell $E_t = [e_1^t, e_2^t, \dots, e_N^t]$ $(N \geq M)$, embedding dimension $d$.
Output: Embedding matrix for each node aligned to the previous embedding space $E_t^* = [e_1^{t*}, e_2^{t*}, \dots, e_N^{t*}]$, where $e_i$ is the embedding for node $i$
Procedure
    $\deg(Node_i|G_{t-1}) = \text{in\_deg}(Node_i|G_{t-1}) + \text{out\_deg}(Node_i|G_{t-1})$
    IND_anchor = [the indices of the nodes with top-$d$ deg $(Node_i|G_{t-1})$]
    $S_t = \left[e^t_{\text{IND\_anchor}_1}, e^t_{\text{IND\_anchor}_2}, \dots, e^t_{\text{IND\_anchor}_d}\right]$ // Construct the subset of $E_t$ with selected nodes
    $S_{t-1} = \left[e^{t-1}_{\text{IND\_anchor}_1}, e^{t-1}_{\text{IND\_anchor}_2}, \dots, e^{t-1}_{\text{IND\_anchor}_d}\right]$ // Construct the subset of $E_{t-1}$ with selected nodes
    $R = argmin_{Q|Q^TQ=I}\|R \cdot (S)^T - (S_{t-1})^T\|_F$ // Solve the optimal alignment mapping
    $E^* = (R \cdot E_t^T)^T$ // Align the embedding space
    return $E_t^* = [e_1^{t*}, e_2^{t*}, \dots, e_N^{t*}]$
```

Following this alignment, we compute the magnitude and rate of semantic displacements (i.e., a word's semantic shift over time-spells) by calculating the cosine distance of a word's embedding at across time-periods. This is done with the following computation:

$$\text{cosine} - \text{dist}(w_t,\ w_{t+\Delta}),$$

where w_t is the location of a word within the embedding space at time t, and $w_{t+\Delta}$ is the location of the same word in the embedding space subsequent to time t. Computing semantic displacement enables the identification of which words are shifting most in their meaning. When applied to hacker forum datasets, this helps pinpoint emerging terminology (e.g., the importance of specific functions, etc.).

5 EVALUATIONS: BENCHMARK DATASETS AND EXPERIMENTS

A key aspect of developing a novel computational approach for a critical cybersecurity application is the rigorous evaluation of the proposed approach against state-of-the-art approaches. Evaluations are typically conducted on a gold-standard, ground-truth dataset. Well-established performance metrics and statistical significance tests are used to ascertain the performance of all algorithms. In accordance with these principles, we carefully design an extensive set of experiments to evaluate the proposed approach. The following sub-sections summarize our data collection, experiments, performance metrics, benchmark methods, and computational setup. While we omit the title of this forum to protect ourselves from hackers within this community, we provide these data to facilitate future scientific research within this area. Interested readers can contact the authors or can directly access this project's GitHub repository at https://github.com/HongyiZhu/D-GEF. This repository also contains all parameters and experiment settings.

5.1 Data Collection

We collect a large and long-standing international hacker forum for analysis. This forum was selected for several reasons. First, it was suggested by several cybersecurity experts who are well versed in Dark Web analytics from academic and industry perspectives. Second, this forum is well known within the online hacker community for being entirely focused on providing malicious tools. This includes numerous zero-day (0-day) tools that have been used for well-publicized, large-scale attacks. Third, and relatedly, no general forum discussions exist; each post is a separate, distinct threat. Consequently, it is ideal for the proposed analysis. Fourth, all forum content is accessible without direct hacker invitations. Finally, this forum has contributors from the Middle East, Russia, U.S., and other geo-political regions.

Collecting hacker forums is a non-trivial technical task. Among other challenges, hacker forums contain significant non-natural language text, include "drive-by malware" that infects users who access the site, block crawling attempts, and often put researchers at risk. Recognizing these issues, we designed a custom Tor-routed web spider to crawl and download all hacker forum HTML pages onto our local hard disks for offline processing. The web spider operated used a breadth first search strategy. Routing traffic through the Tor network enabled us to maintain anonymity against hackers within these communities [24]. A specialized Python program using Regular Expressions parsed all data into a relational database.

Our collection procedures resulted in a dataset with 32,766 posts (i.e., threats) made by 8,429 hackers between January 1, 1996 and July 10, 2019 (23-year period). Since the forum is designed for providing hackers a platform to share high-quality threats, commonly available attributes include title, description, full source code, post date, target platform, attack type, and author name. To support the proposed evaluation procedure (summarized in the following sub-sections), we used the attack type and target platform attributes to develop a gold-standard dataset. Each must be verified (e.g., executed) by another forum member before the post is made public on the forum. Therefore, they are the most trustworthy labels in the collected data to establish ground truth.

Both attributes were used to retrieve remote and local exploits to comprise our gold-standard dataset. Remote exploits are those that require a network connection to be executed and/or no prior access to a machine has been attained. Local exploits operate after an attacker has gained prior access to a system. These exploits were retrieved for two reasons. First, they are the most prevalent within our datasets, and thus can provide large corpora to generate embeddings and conduct subsequent evaluations. Second, the exploit's most common target platforms are Windows

Table 4. Summary of Hacker Forum Testbed Used for Benchmark Experiments

Threat Type	Target Platform	Date Range	# of Exploits	Total
Remote	Windows	03/23/2009 – 07/05/2019	1,418	**1,864**
	Linux	06/24/2000 – 07/02/2019	446	
Local	Windows	09/28/2004 – 06/20/2019	1,818	**2,429**
	Linux	01/01/1996 – 07/02/2019	611	
Total:	–	**01/01/1996 – 07/05/2019**	**4,293**	**4,293**

Table 5. Summary of Graphs of Words Generated from Hacker Forum Testbed

Dataset	Topology			In-degree			Out-degree		
	Size	# of nodes	# of edges	Min	Max	Avg.	Min	Max	Avg.
Local Linux	611	1,087	1,615	0	209	1.486	0	80	1.486
Local Windows	1,818	2,359	4,221	0	227	1.789	0	121	1.789
Remote Linux	446	862	1,219	0	214	1.414	0	29	1.414
Remote Windows	1,418	2,061	3,653	0	267	1.772	0	106	1.772
All	**4,293**	**5,142**	**9,935**	**0**	**484**	**1.932**	**0**	**162**	**1.932**

and Linux (as listed by the forum). Both are widely deployed in enterprise and other contexts. Table 4 summarizes the date range and number of exploits for each category.

Overall, the gold-standard dataset contains 4,293 exploits (1,864 remote and 2,429 local) made between January 1, 1996 and July 5, 2019. The number of Windows exploits totals 3,236, while the total of the Linux exploits is 1,057. All exploits are non-overlapping; no exploit appears in more than one category. To provide further granularity, we report key summary statistics for each dataset's GoW in Table 5. In particular, we present the size (i.e., number of threat posts); number of nodes; number of edges; and minimum, maximum, and average in-degree and out-degree.

Overall, the number of nodes across graphs ranged from 862 to 2,061, while the number of edges ranged from 1,219 to 4,221. Examining the minimum, maximum, and average in-degree and out-degree values indicates that each dataset follows a power law distribution, wherein most nodes have a low centrality, but a select few have values significantly above the average. This indicates that the distribution seen in the gold-standard dataset represents to many others seen in related hacker forum literature pertaining to network science and embedding analysis [7, 43].

5.2 Experimental Procedure: Objectives and Performance Metrics

D-GEF is fundamentally an unsupervised approach to generating and mapping word embeddings across multiple semantic spaces. Although possessing significant descriptive capabilities, the diachronic component cannot be directly evaluated, unless ground truth about the precise semantic shifts are known. Our scenario is like many other recent diachronic linguistics studies, wherein our dataset does not offer this knowledge. Evaluations in these literature focus on evaluating the quality of the generated word embedding.

Two major evaluation approaches for word embeddings exist: extrinsic and intrinsic [4]. Extrinsic evaluations input the proposed algorithm's generated word embeddings into a selected downstream task (e.g., classification). Performance is measured against alternative approaches for generating embeddings. Intrinsic evaluations directly examine word embedding quality. Common tasks include semantic analogy and cluster purity [47]. In this study, we carefully design four sets of experiments to identify the ideal graph embedding approach. Table 6 presents a summary of the experiment type, experiment, and evaluation metrics.

Table 6. Summary of Benchmark Experiments

Experiment Type	Experiment*	Evaluation Metrics
Intrinsic	1: Semantic Analogy	Accuracy
	2: Cluster Purity	Homogeneity, completeness, v-measure, Adjusted Mutual Information (AMI), Adjusted Rand Index (ARI)
Extrinsic	3: Target Platform Classification	Accuracy, precision, recall, F-Measure, Receiver Operating Characteristics (ROC) curve, Area Under the Curve (AUC)
	4: Attack Type Classification	

**Note*: Consistent with prior literature, the entire ground-truth dataset is used for all experiments.

Table 7. Summary of Analogies Used for Experiment 1

	Local-Linux	Local-Windows	Remote-Linux	Remote-Windows	All
Vendor-Program	153	325	231	630	4,005
Object-Action	55	66	120	171	1,275
Total:	**208**	**391**	**351**	**801**	**5,280**

The entire ground-truth dataset, irrespective of the time period the posts they appear in, is used for the proposed intrinsic and extrinsic experiments. Benchmarking in this fashion is commonly accepted practice in computational diachronic linguistics literature. The following sub-sections describe each experiment.

5.2.1 Experiment 1: Semantic Analogy (Intrinsic). Experiment 1 is an intrinsic evaluation that evaluates semantic analogy of words that appear in the graph representation [54]. Proposed from word embedding literature, the intuition of this evaluation is that a good low-dimensional embedding should preserve the (geometric) semantic relationships between high-dimensional words. In particular, assuming word pairs (A, B) and (X, Y) have the same semantic relationship, a word analogy task aims to evaluate a word embedding model's ability to correctly infer Y when A, B, and X are given. For example, a good embedding model should capture the "Vendor-Program" relation from ("Google," "Chrome"), and infer "Firefox" when given ("Mozilla," ?). The inference is equivalent to searching the best word Y that minimizes the following total distance $Dist_{A,B,X}$:

$$Y = \arg\min_{\mathrm{i}} Dist_{A,B,X} = \arg\min_{i} dist\,(B, i) + dist\,(C, i) - dist\,(A,\ i)\,.$$

A cybersecurity expert well-versed in Dark Web analytics and threat intelligence examined our dataset and defined two relationships for evaluation: "Vendor-Program" (e.g., ("Google," "Chrome"; "Mozilla," "Firefox")) and "Object-Action" (e.g., ("command," "execution"; "credentials," "disclosure")). Based on these, we manually defined analogies for each dataset and category. Adhering to best practices, we asked a panel of three cybersecurity students and one professor to validate the exploits in each category. To prevent any biases (e.g., social desirability), we asked each panelist to perform this task independent of other participants. We calculate the level of agreement between the raters using the Cohens' Kappa statistic, reaching a final value of 0.9978. All analogies that were disagreed upon were omitted from the dataset. Table 7 presents a summary of the analogies for each dataset.

Overall, the resultant analogy quantities were 208 for Local Linux, 391 for Local Windows, 351 for Remote Linux, and 801 for Remote Windows with a total of 5,280 overall. These quantities are consistent with past literature executing word embedding evaluation tasks. To computationally implement the proposed evaluation, each embedding model generates the top-5 candidate words of Y when given A, B, and X. A hit is recorded if the top-5 candidates contain the correct Y. Accuracy evaluates analogy task performance by:

$$\text{Accuracy} = \frac{\#\text{ of hits}}{\#\text{ of all (A, B; X, Y) tuples}}.$$

This evaluation was run for all methods for each dataset individually. We also ran the analogy task for the overall dataset (i.e., all 5,280 analogies). Benchmarking in this fashion is commonly accepted practice in word embedding related literature.

5.2.2 Experiment 2: Clustering (Intrinsic). These unique domain characteristics motivate the second intrinsic task in which we evaluate the D-GEF's word embeddings: k-means clustering. k-means is a popular clustering algorithm commonly used in differential privacy [50] and malware analysis tasks [53] as it provides easily interpretable results in a computationally efficient manner [21]. In this study, identifying the semantic displacement of a word requires computing the distance a word travels across multiple embedding spaces. A high-quality embedding is needed to ensure accurate distance calculations. This is also true for k-means. If embeddings are lower-quality, then similar entities in the ground-truth data will have a larger distance calculated between them and will be clustered apart, and overall clustering performance will suffer. This intuition has made k-means a popular approach to evaluating word embedding quality [4, 61]. Moreover, CTI professionals, such as Security Operations Center (SOC) analysts, often wish to comprehensively understand an exploit's functions and implementation by examining closely associated terms [9, 14, 49]. This knowledge can pinpoint new exploit names and emerging trends (e.g., Mirai for DDoS).

The top-200 keywords with the highest word frequencies were identified in each of the four sub-datasets of Local Linux, Local Windows, Remote Linux, and Remote Windows. Duplicated keywords that showed up in two or more datasets were removed, resulting in 118, 97, 107, and 89 unique keywords for four categories, respectively. The clustering task is then conducted as follows:

- **Step 1:** The word embedding model produces word vectors for all the keywords.
- **Step 2:** A clustering algorithm (e.g., the k-means algorithm) is used to separate the word vectors into $k (k = 4)$ categories.
- **Step 3:** Clustering result evaluation.

Five well-established performance metrics evaluate the quality of clustering results: Adjusted Mutual Information (AMI), Adjusted Rand Index (ARI), Completeness, Homogeneity, and V-Measure. AMI measures the quantity of mutual information available between clustering results and the ground truth. ARI computes a similarity measure between clusters by considering all pairs of samples and counting how many pairs are assigned to clusters in the predicted vs. ground truth. Completeness identifies if nodes within the pre-defined cluster in the gold-standard dataset appear in the same cluster. Homogeneity identifies how many word embeddings in each cluster have the same label. V-Measure is the harmonic mean of completeness and homogeneity [37]. Each metric calculates a scalar value between 0.0 (random assignment) and 1.0 (perfect match).

5.2.3 Experiment 3: Exploit Target Platform Classification (Extrinsic). Experiment 3 is an extrinsic evaluation task that focused on identifying how the embeddings generated from each benchmark method performed in a binary classification task. In particular, we set the output label as platform the exploit is targeted at (i.e., Windows or Linux). This evaluates an embedding's ability to discern what system an exploit is designed for (a common task for security analysts). The classification task was executed as follows: First, we adopted all aforementioned pre-processing, representation, and embedding algorithmic procedures to generate word embeddings. Second, word embeddings within each post are summed and inputted into an SVM classifier. Finally, each post is classified based on the summed representation. Classifying posts in this fashion is commonly accepted practice in related literature.

The entire ground-truth dataset is used for this evaluation task. Since the dataset is imbalanced (3,236 Windows, 1,057 Linux), each algorithm was trained and tested using a 10-fold cross validation (CV) strategy. We evaluate algorithm performances using well-established metrics of accuracy, precision, recall, and F1-score (i.e., F-measure). Each uses a combination of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) to compute the metrics. The formulations are presented below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \qquad Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN}, \qquad F1 - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}.$$

Given the imbalanced nature of the ground-truth dataset, we also use two additional performance metrics to evaluate classification performance: Receiver Operating Characteristics (ROC) curve and Area Under the Curve (AUC). ROC uses a 2-dimensional space to plot the false positive rate (x-axis) and true positive rate (y-axis). This curve can be used to calculate the AUC score. AUC quantifies the tradeoffs between type I and type II errors by computing a scalar metric ranging from 0.5 (random guess) and 1.0 (perfect performance).

5.2.4 Experiment 4: Attack Type Classification (Extrinsic). Experiment 4 aimed to examine how the embeddings generated from each benchmark method performed in a binary classification task when the output label was the exploit type (i.e., remote or local). Delineating between exploits is a critical CTI task. Like Experiment 3, each post's word embeddings were summed and inputted into an SVM classifier. Performance metrics of accuracy, precision, recall, and F1 were calculated. Since the ground-truth dataset was also imbalanced in this setting, AUC and ROC were calculated.

5.3 Benchmark Methods

For each experiment, we selected prevailing word and graph embedding approaches to evaluate. Word embedding models aim to map words into an embedding by operating a series of mathematical functions upon a corpus. These functions can use neural network, dimensionality reduction, probabilistic, and other techniques to create embeddings. For this research, we evaluate six prevailing models that rely on shallow neural networks (most closely related) to generate an embedding. They are as follows:

- *word2vec with Continuous Bag-of-Words (CBOW)* aims to reconstruct linguistic contexts of words by predicting a word based on a surrounding window [29].
- *word2vec with Skip-Gram with Negative Sampling (SGNS)* uses one word to predict surrounding words.
- *doc2vec with Distributed Bag-of-Words (DBOW)* enforces that the model predicts words randomly sampled from a paragraph without providing any surrounding context.
- *doc2vec with Distributed Memory (DM)* randomly samples consecutive words from a paragraph and aims to predict a center word from the sample based on its context.
- *fastText with CBOW* is like word2vec CBOW, but accounts for character and word n-grams.

- *fastText with SGNS* operates like word2vec SGNS, but accounts for character and word *n*-grams.

Graph embedding methods include those that are random walk-based (node2vec, DeepWalk), deep representation learning (SDNE), graph factorization (GF, HOPE, GraRep), and edge reconstruction (LINE). All operate on the same GoW representation. Taken together, the selected graph embedding models represent the prevailing unsupervised approaches operating on directed graphs to produce task-independent embeddings suitable for selected downstream tasks (e.g., diachronic linguistics). We also evaluate the classification tasks with the prevailing non-embedding approach, TF-IDF.

5.4 Computational Setup

All experiments were conducted using the Python programming language, version 3.7. All word embedding approaches were implemented using the Genism package. Graph embedding methods were implemented using the Open Network Embedding (OpenNE) package. Performance metrics and statistical tests were conducted using scikit-learn. To help facilitate fair comparisons, all embeddings were 128 dimensions. All experiments were conducted on a single workstation equipped with an Intel® Core i7-8550U @1.80GHz processor, 16 GB of RAM, and an NVIDIA® GeForce MX150 Graphical Processing Unit (GPU) with 2 GB of onboard RAM. All methods used default parameters, were trained over 20 epochs, and generated 128 dimension embeddings. We also provide the entire experimental framework in this project's publicly accessible GitHub repository such that interested readers can reproduce the presented results. All hyperparameters for the selected approaches are accessible on this page.

6 EXPERIMENT RESULTS

The results for each experiment are summarized in the following sub-sections. For space considerations, we group the results of the intrinsic evaluations (Experiments 1 and 2) in Section 6.1 and 6.2, and the extrinsic evaluations (Experiments 3 and 4) in Section 6.3. Within each sub-section, we discuss the overall observations and key takeaways from the experiment.

6.1 Results of Experiments 1: Semantic Analogy

Table 8 presents the results of the analogy and clustering evaluations. Results are grouped based on method category (i.e., word embedding, graph embedding). Accuracy is computed for each dataset as well as across datasets. The top performing algorithm is highlighted in boldface.

For the Local Windows, Remote Linux, and Remote Windows datasets, the conventional graph factorization approach outperformed all other categories of methods. All graph embedding methods except HOPE outperformed all word embedding approaches. For the Local Linux dataset, LINE achieved the highest accuracy at 0.317. When accounting for all data, node2vec attained the highest performance at 0.253. the second and third best approaches were also graph embedding algorithms, GF, and LINE, respectively.

Overall, the consistency of these results across the multiple, disparate datasets suggests that the underlying graph of words representation assisted in capturing local and global relationships within the threat corpora that are missed by prevailing word embedding models. A possible explanation for graph factorization consistently achieving the best performance is that it focuses on primarily capturing local proximities. Other graph embedding methods may aim to capture both types of proximities (e.g., SDNE). In a task such as semantic analogy, wherein proximity of wording is often critical for establishing analogical relationships, such computations may weight critical semantic cues incorrectly.

Table 8. Summary of Benchmark Experiment 1 Results

Embedding Category	Type	Method	Dataset				
			Local Linux	Local Windows	Remote Linux	Remote Windows	All
Word	word2vec	SGNS	0.014	0.049	0.006	0.014	0.027
		CBOW	0.010	0.018	0.000	0.006	0.007
	fastText	SGNS	0.024	0.008	0.023	0.007	0.011
		CBOW	0.010	0.003	0.014	0.000	0.009
	doc2vec	DM	0.010	0.018	0.000	0.007	0.023
		DBOW	0.005	0.056	0.006	0.015	0.031
Graph	Graph Factorization	GF	0.250	**0.161**	**0.316**	**0.203**	0.237
		HOPE	0.019	0.000	0.020	0.007	0.002
		GraRep	0.014	0.010	0.031	0.009	0.005
	Random Walk-based	DeepWalk	0.308	0.130	0.197	0.191	0.248
		node2vec	0.308	0.059	0.111	0.124	**0.253**
	Deep Representation Learning	SDNE	0.019	0.003	0.000	0.006	0.004
	Edge Reconstruction	LINE	**0.317**	0.118	0.145	0.195	0.153

Table 9. Summary of Benchmark Experiment 2 Results

Method Category	Type	Method	Clustering				
			AMI	ARI	Completeness	Homogeneity	V-Measure
Word	word2vec	SGNS	0.236	0.205	0.269	0.222	0.243
		CBOW	0.101	0.088	0.130	0.096	0.110
	fastText	SGNS	0.188	0.163	0.203	0.187	0.195
		CBOW	0.046	0.031	0.058	0.051	0.054
	doc2vec	DM	0.112	0.095	0.142	0.105	0.121
		DBOW	0.265	0.201	0.292	0.253	0.271
Graph	Graph Factorization	GF	0.249	0.196	0.262	0.248	0.255
		HOPE	0.048	0.027	0.076	0.046	0.057
		GraRep	0.121	0.074	0.138	0.121	0.129
	Random Walk-based	DeepWalk	**0.333**	0.270	**0.390**	0.300	**0.340**
		node2vec	0.307	**0.281**	0.316	**0.310**	0.313
	Deep Representation Learning	SDNE	0.076	0.069	0.119	0.067	0.085
	Edge Reconstruction	LINE	0.258	0.236	0.277	0.253	0.265

6.2 Results of Experiment 2: Clustering Threat Types

Table 9 presents the results of the clustering evaluations. Results are grouped based on method category (i.e., word embedding, graph embedding). AMI, ARI, Completeness, Homogeneity, and V-Measure are computed for each dataset. The top-performing algorithm is highlighted in boldface.

Across the board, the random walk-based methods of DeepWalk and node2vec outperformed all other methods. In terms of V-Measure (overall measure of clustering performance), DeepWalk achieved the highest performance of 0.340. DeepWalk also attained the highest performance in AMI (0.333) and Completeness (0.390), while node2vec had the highest scores in ARI (0.281) and Homogeneity (0.310). As in Experiment 1, nearly all graph embedding approaches outperformed the conventional word embedding approaches. Doc2vec with DBOW and word2vec with SGNS were the only exceptions, with the methods achieving V-Measure scores of 0.271 and 0.243, respectively.

Overall, these results indicate that the graph embedding approaches better captured words within the same category (i.e., intra-cluster purity), and were able to more accurately distinguish words across categories (i.e., inter-cluster purity). The overall performance in AMI and ARI suggests that the graph embedding approaches better captured the underlying distribution of word categories. Relatedly, the completeness indicates that the proposed representation can map more

Table 10. Summary of Results from Benchmark Experiments 3 and 4

Experiment 3 Results Attack Type (Local vs Remote) Classification							
Embedding Category	Operation	Method	Performance Metrics				
			Accuracy	Precision	Recall	F1	AUC
Non-embedding	TF-IDF	χ^2	87.6±1.3%	**88.0±2.6%**	77.4±1.9%	80.9±2.0%	0.872
		F-value	87.4±1.2%	87.7±2.6%	77.1±1.9%	80.5±1.9%	0.877
		Mutual Info	84.1±1.2%	84.4±2.8%	70.5±2.0%	73.8±2.1%	0.818
Word	word2vec	SGNS	84.9±1.4%	82.2±2.5%	74.2±2.6%	76.9±2.6%	0.898
		CBOW	83.5±1.1%	81.5±1.5%	70.7±2.3%	73.6±2.3%	0.869
	fastText	SGNS	85.9±1.5%	83.6±2.4%	76.3±2.5%	78.9±2.4%	0.907
		CBOW	83.2±1.4%	80.3±2.4%	70.6±2.2%	73.4±2.4%	0.868
	doc2vec	DM	82.3±1.5%	79.9±2.0%	68.3±2.5%	71.0±2.8%	0.846
		DBOW	84.7±1.3%	81.9±2.0%	74.2±2.2%	76.8±2.1%	0.899
Graph	Graph Factorization	GF	87.4±1.1%	85.2±1.6%	79.1±2.2%	81.4±2.0%	0.914
		HOPE	84.2±1.2%	82.3±1.2%	72.3±2.9%	75.1±2.8%	0.884
		GraRep	86.7±1.4%	83.9±2.1%	78.4±2.7%	80.5±2.5%	0.914
	Random Walk-based	DeepWalk	**88.1±0.8%**	86.2±1.7%	80.5±1.6%	**82.7±1.1%**	**0.926**
		node2vec	88.0±1.4%	85.7±2.1%	**80.6±2.7%**	82.6±2.3%	0.924
	Autoencoder	SDNE	83.8±1.7%	81.2±2.2%	71.8±3.2%	74.6±3.3%	0.875
	Edge Reconstruction	LINE	86.2±1.4%	83.5±2.2%	77.2±2.9%	79.5±2.6%	0.913

Experiment 4 Results: Platform (Linux vs Windows) Classification							
Embedding Category	Type	Method	Performance Metrics				
			Accuracy	Precision	Recall	F1	AUC
Non-embedding	TF-IDF	χ^2	84.7±1.8%	**88.0±2.6%**	77.4±1.9%	80.9±2.0%	0.914
		F-value	85.1±1.9%	87.7±2.6%	77.1±1.9%	80.5±1.9%	0.913
		Mutual Info	81.7±1.2%	84.4±2.8%	70.5±2.0%	73.8±2.1%	0.890
Word	word2vec	SGNS	85.3±1.7%	82.2±2.5%	74.2±2.6%	76.9±2.6%	0.921
		CBOW	80.9±1.3%	81.5±1.5%	70.7±2.3%	73.6±2.3%	0.895
	fastText	SGNS	85.0±2.0%	83.6±2.4%	76.3±2.5%	78.9±2.4%	0.921
		CBOW	79.9±1.8%	80.3±2.4%	70.6±2.2%	73.4±2.4%	0.883
	doc2vec	DM	80.2±1.3%	79.9±2.0%	68.3±2.5%	71.0±2.8%	0.882
		DBOW	85.6±1.9%	81.9±2.0%	74.2±2.2%	76.8±2.1%	0.922
Graph	Graph Factorization	GF	86.8±1.2%	85.2±1.6%	79.1±2.2%	81.4±2.0%	0.937
		HOPE	84.2±1.6%	82.3±1.2%	72.3±2.9%	75.1±2.8%	0.921
		GraRep	86.5±1.5%	83.9±2.1%	78.4±2.7%	80.5±2.5%	0.937
	Random Walk-based	DeepWalk	87.1±1.6%	86.2±1.7%	80.5±1.6%	**82.7±1.1%**	0.938
		node2vec	**87.8±1.8%**	85.7±2.1%	**80.6±2.7%**	82.6±2.3%	**0.941**
	Autoencoder	SDNE	83.0±1.3%	81.2±2.2%	71.8±3.2%	74.6±3.3%	0.917
	Edge Reconstruction	LINE	86.4±1.3%	83.5±2.2%	77.2±2.9%	79.5±2.6%	0.937

words in the same categories (from the same sub dataset) to a close region in the lower embedding space. A possible explanation for the strong performance of random walk-based methods retains sequential dependency (1st order proximity). In contrast, competing graph embedding methods capture higher order proximities at a global level, or may miss some of these direct proximities altogether (e.g., word embedding models).

6.3 Results of Experiments 3 and 4: Classification

Table 10 presents the classification evaluation results. Results are grouped based on method category. For each algorithm, we report the accuracy, precision, recall, and F1 scores with a confidence interval. We also present the ROC curves and AUC values in Figure 4. The top-performing algorithm is highlighted in boldface.

Similar to experiment 2, results across both classification tasks indicate that the random walk-based methods outperform the competing graph and word embedding approaches. In terms of F1, the harmonic mean of precision and recall, DeepWalk achieved a score of 82.17%. When examining

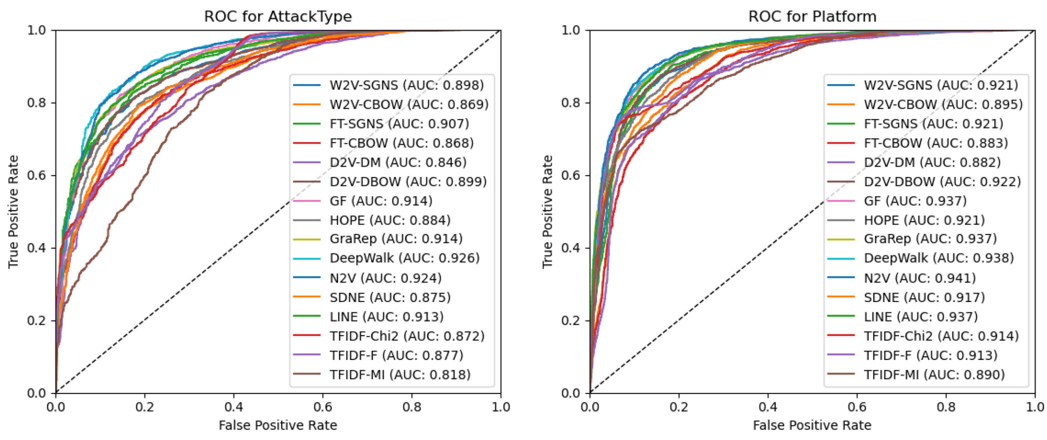


Fig. 4. ROC Curves for Attack Type and Platform Classification Experiments.

Table 11. Summary of Threat Testbeds Used for Case Studies

Threat Type	Targeted Platform(s)	Date Range	# of Posts	# of Authors
Web Application	PHP	1/1/2012 – 7/8/2019	6,577	1,461
DoS	Windows	1/1/2012 – 6/26/2019	1,355	376
Total:	–	**1/1/2012 – 7/8/2019**	**7,932**	**1,837**

AUC (critical for evaluating unbalanced datasets), DeepWalk and node2vec attained the highest values on both datasets across all benchmark methods.

As with the first two experiments, the graph embedding approaches consistently outperformed the conventional word embedding techniques. A differentiator between the intrinsic and extrinsic experiments is that the former only examined the quality of the word embeddings directly. However, the extrinsic evaluations examined how the word embeddings when aggregated together can outperform the conventional approaches. Taken together, the consistency of the results across multiple datasets and evaluation tasks suggest that the graph embedding approach captures local and global relationships of words missed by prevailing word embedding methods.

7 CASE STUDY: PRACTICAL UTILITY AND SECURITY IMPLICATIONS

In this section, we demonstrate the potential practical utility and security implications of the proposed D-GEF. To facilitate this demonstration, we identified two additional threat categories from our collected forum (summarized in Section 5.1). The first category is web application threats that target PHP technologies. The second is denial of service (DoS) threats that target the Windows operating system. Both threats are retrieved using the available metadata of target platform and attack type. Both were selected as they are common threats against prevailing and widely used technologies. Table 11 summarizes each threat type, the targeted platform, date range, number of posts (i.e., threats), and number of authors. To control the scope of these studies, we limit our analysis to the past eight years (2012 - 2019).

In total, our testbed contains 7,932 posts (i.e., threats) made by 1,837 authors. Web applications encompassed 6,557 threat posts made by 1,461 authors, while the DoS component comprised of 1,355 posts made by 376 authors. No posts were used in the benchmark experiments. Consequently, they are "wild" posts, wherein we do not perform any annotation or have any prior knowledge about the dataset. All posts are non-overlapping. Given the disparities in size, scale, and topical coverage, we process each threat type as a separate case study. The steps of time-spell specification, text graph formulation, descriptive statistics calculation, and detection of semantic shifts used to execute each study simulates the process a CTI professional can take when aiming to identify threat trends and terms.

The following sub-sections present the results of executing this process on both datasets. We note that the presented analysis illustrates only a selection of possibilities attainable. Undoubtedly, there are numerous other variations and specifications available to end-users operating this framework. It is not our goal to enumerate all options; rather, it is to illustrate selected possibilities. Section 7.4 discusses potential security implications of the results and promising future directions to expand the presented analysis.

7.1 Time-Spell Specification and Text Graph Formulation

Conducting diachronic analysis requires the dataset in question to be split into multiple time-spells [19]. As such, we split our dataset into time intervals of three months (four quarters per year). This breakdown is consistent with the analysis timeframes used by many industry CTI reports. To counter the issues of simple term frequency-based approaches, we create text graphs at each time-spell. Each graph builds upon the previous. Nodes represent all words in threat posts in that time-spell, and edges denote if two words appeared in the same post. Table 12 presents selected year-end visualizations of the generated GoWs. Node size represents degree score. Such visualizations are commonly displayed in graph-based security analytics and CTI literature [43]. For space considerations, we only present the year-end results for four years.

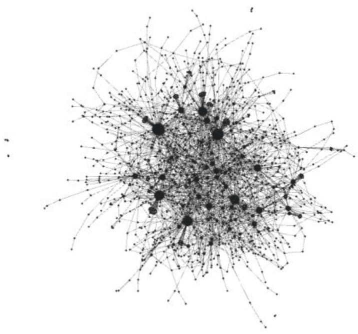
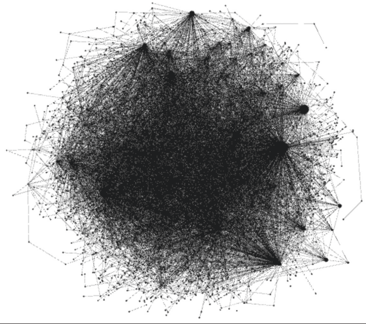
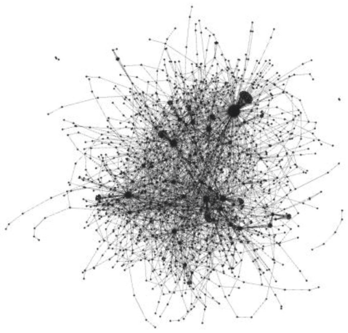
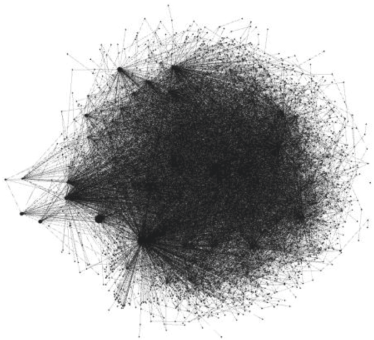
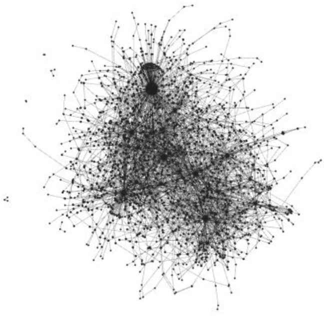
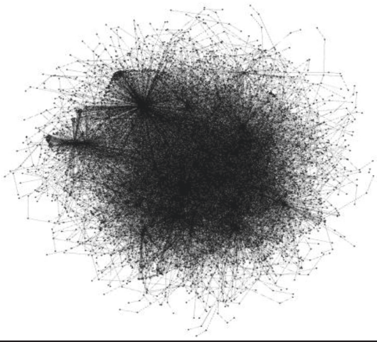
For both threat sets, the visualizations reveal that the graphs are expanding over the specified time-period. This indicates that the lexicon and vocabulary of hacker threats are expanding over time. Between the two, the visualizations indicate that the DoS threat landscape is evolving more rapidly and quickly than the web applications. Such insights can offer significant value for SOC analysts and related cybersecurity professionals in prioritization and related mitigation activities. However, as indicated in our methodological review, representing text as a GoW can provide access to metrics that can further reveal the diversity, breadth, and characteristics. To this end, the following sub-section summarizes the selected metrics we calculated for each threat type.

7.2 Descriptive Statistics: Node and Global Metrics

We opted to calculate node and network level measures to understand local and global network dynamics, respectively. Node level metrics include the minimum, maximum, and average in-degree and out-degree. Network level metrics include number of nodes, number of edges, diameter, radius, density, average path length, average clustering, and average eccentricity. Each was carefully selected based on their ability to reveal various insights about the key terms, growth, and breadth of threat vocabularies. Results are summarized in Table 13. For space considerations, we only list year-end results. The numbers in each time-spell account for the total in previous spell(s).

Table 12 reveals several key insights unavailable from the graph visualizations. First, the surges seen in the number of posts were met with a commensurate increase in the number of nodes (i.e., words). This growth indicates that hackers are using a richer lexicon. The increase in vocabulary size was concurrent with the decrease in overall graph density and average clustering coefficient. The decrease in both measures indicates that hackers diversify their interests and specialties, resulting in new threats implementing novel functionalities. The increases in average path length and network diameter support this observation. The rate of changes for each of these metrics varied across the different threat types. In the case of web application threats, the rates plateaued around 2017. In contrast, the DoS threats saw more rapid rates of development and expansion throughout the specified time-periods. Taken together, these metrics suggest that DoS threats are

Table 12. Selected Year-End Visualizations of Threat GoWs

Year-End	GoWs for DoS Threats	GoW Visualizations for Web Application Threats
2017		
2018		
2019		

growing more varied, while web applications ones have relatively stabilized. This insight is critical; organizations have limited ability to mitigate all threats. Therefore, carefully prioritizing threats using such metrics is essential.

Examining the node level statistics reveals that each graph follows a power-law degree distribution, wherein a few nodes have an above-average centrality, and the majority of nodes fall below the average. For both datasets, the maximum in-degree and out-degree increased each year. This indicates that the new threats being introduced relied on a selected range of core features and functionalities. To gain insight into these key words in the network, we summarize the top

Table 13. Topological and Node Level Descriptive Statistics Between 2012-2019

Network Level and Node Level Descriptive Statistics for Web Application Threats									
Category	Metric	2012	2013	2014	2015	2016	2017	2018	2019
Forum	# of threats	1,742	2,429	3,015	3,727	4,261	5,383	6,250	6,577
Network Level Metrics	# of nodes	2,310	3,003	3,515	4,058	4,436	5,229	5,853	6,065
	# of edges	3,575	4,950	6,075	7,411	8,353	10,311	11,860	12,389
	Diameter	16	17	17	15	15	15	15	15
	Radius	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Density	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
	Avg. path length	5.336	6.077	5.632	5.386	5.345	5.200	5.135	5.120
	Avg. clustering	0.012	0.014	0.014	0.015	0.015	0.015	0.015	0.014
	Avg. eccentricity	10.710	12.184	11.945	10.617	10.657	10.684	10.690	10.682
Node Level Metrics	Min. in-degree	0	0	0	0	0	0	0	0
	Max in-degree	460	557	627	705	750	931	1,115	1,184
	Avg. in-degree	1.548	1.648	1.728	1.826	1.883	1.972	2.026	2.043
	Min. out-degree	0	0	0	0	0	0	0	0
	Max out-degree	67	113	167	295	347	443	481	494
	Avg. out-degree	1.548	1.648	1.728	1.826	1.883	1.972	2.026	2.043
Network Level and Node Level Descriptive Statistics for DoS Threats									
Category	Metric	2012	2013	2014	2015	2016	2017	2018	2019
Forum	# of threats	199	306	359	512	708	938	1,197	1,355
Network Level Metrics	# of nodes	402	557	632	865	1,158	1,423	1,699	1,838
	# of edges	514	763	880	1,277	1,824	2,356	2,876	3,188
	Diameter	12	17	16	21	20	19	22	23
	Radius	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	Density	0.003	0.002	0.002	0.002	0.001	0.001	0.001	0.001
	Avg. path length	3.889	6.144	6.021	7.215	6.678	6.733	7.029	7.029
	Avg. clustering	0.029	0.026	0.026	0.027	0.030	0.030	0.024	0.023
	Avg. eccentricity	4.953	9.673	8.975	10.612	11.331	11.248	14.623	15.712
Node Level Metrics	Min. in-degree	0	0	0	0	0	0	0	0
	Max in-degree	38	49	53	63	82	109	217	287
	Avg. in-degree	1.279	1.370	1.392	1.476	1.575	1.656	1.693	1.734
	Min. out-degree	0	0	0	0	0	0	0	0
	Max out-degree	13	18	19	24	29	49	62	67
	Avg. out-degree	1.279	1.370	1.392	1.476	1.575	1.656	1.693	1.734

ten words based on degree and betweenness centralities in Table 14. For space considerations, we only list the ranks of words at the year-ends of 2017 – 2019.

The top ten words in terms of degree centrality remain relatively unchanged at each year-end for both threat types. This indicates that even if new threat types emerge, the core functions of the threats remain the same. In the case of web applications, common terms include PHP-based web application technologies being targeted such as "WordPress" and "Joomla" or specific exploit functions such as "Remote," "Script," "Arbitrary," or "Injection." Each pertains to a common web exploit family, including SQL injection, cross-site scripting, and others. In the case of DoS, top words in terms of degree and betweenness pertain to the specific target within the Windows system the threat is aiming to attack. This includes "Memory," "Buffer," and "Stack."

The high degree and betweenness values for each of these terms indicate that the entire threat vocabulary is dependent upon these key aspects to maintain cohesiveness; without them, the entire network could easily fall apart. CTI professionals can use this knowledge to discern between threat types, capabilities, and operations (knowledge often gleaned by malware analysis). Moreover, these words indicate the specific targets that require the highest level of protection. While useful, the intelligence provided by Table 14 does not reveal the specific features incorporated

Table 14. Topological and Node Level Descriptive Statistics between 2017-2019

Top Degrees of Terms for Web Application Threats									
Deg. Rank	2017			2018			2019		
	Word	Deg.	Between. (rank)	Word	Deg.	Between. (rank)	Word	Deg.	Between. (rank)
1	Injection	986	3	Injection	1175	3	Injection	1,246	2
2	Multiple	474	6	WordPress	494	1	WordPress	507	1
3	WordPress	456	1	Multiple	481	7	Multiple	495	7
4	Cross	399	16	Cross	447	18	Cross	454	19
5	CSRF	327	4	Plugin	353	5	Plugin	374	4
6	Plugin	299	5	CSRF	334	4	Arbitrary	341	13
7	Arbitrary	291	14	Arbitrary	330	13	CSRF	338	5
8	Remote	255	13	Joomla	301	6	Joomla	311	6
9	Joomla	230	7	Component	298	11	Component	311	10
10	Script	219	9	Remote	280	14	Remote	301	15
Top Degrees of Terms for DoS Threats									
Deg. Rank	2017			2018			2019		
	Word	Deg.	Between. (rank)	Word	Deg.	Between. (rank)	Word	Deg.	Between. (rank)
1	Denial	111	4	Denial	219	4	Denial	289	4
2	Buffer	102	9	Buffer	110	12	Buffer	113	10
3	Server	86	7	Exploit	102	322	Exploit	113	14
4	Exploit	84	276	Server	98	6	Server	112	6
5	Memory	80	3	Memory	85	3	Memory	86	3
6	Crash	77	25	Crash	77	31	Crash	77	42
7	Windows	56	1	Windows	69	1	Windows	74	1
8	Stack	49	16	UseAfterFree	53	17	UseAfterFree	57	15
9	UseAfterFree	48	19	Stack	51	18	Local	55	18
10	Player	47	18	Local	50	15	Remote	52	5

over the years to develop new web applications or DoS threats. We gain deeper insight into this by employing diachronic semantic displacement calculations to compute the average amount a word shifts between time-spells. We present these results next.

7.3 Diachronic Analysis: Detection of Semantic Shifts

For the purpose of these case studies, we employ the DeepWalk algorithm to generate embeddings. DeepWalk was selected as it consistently achieved strong performances across the four benchmark experiments. Moreover, it can operate upon the specified graph. Within the D-GEF framework, DeepWalk is applied to create embeddings in each time-spell. We then align embedding spaces using the Frobenius norm specification. Subsequently, we calculate each word's semantic displacement (i.e., how much words shift in their meaning) to identify emerging threat trends. All processes are fully automated; no manual intervention is required at any stage of the framework. The top 20 words with the highest average shift are summarized in Table 15. Words are presented in their raw format.

Semantic shift values indicate how much the meaning has shifted from its original meaning. Larger values indicate that term has shifted further away from its original meaning. Each word shifted an average amount of 5.637 for web applications and 4.421 in semantic space for the DoS threats. Words with the top 20 average shifts relate to specific functionalities. For web applications, this included "sqli" for SQL injection, "permission" for gaining access to specific file paths

Table 15. Top Shifted Words Between 2012-2019 (*average shift per time-spell)

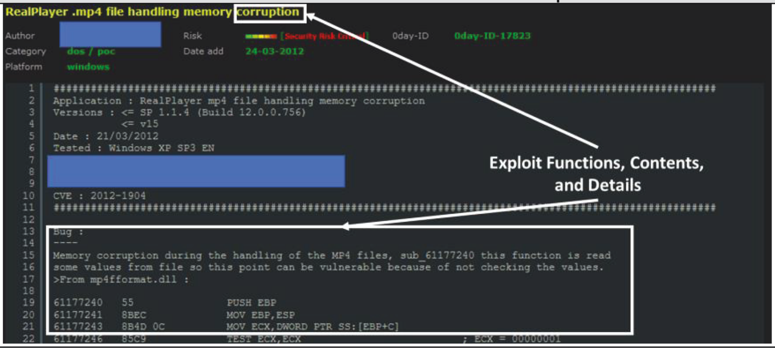
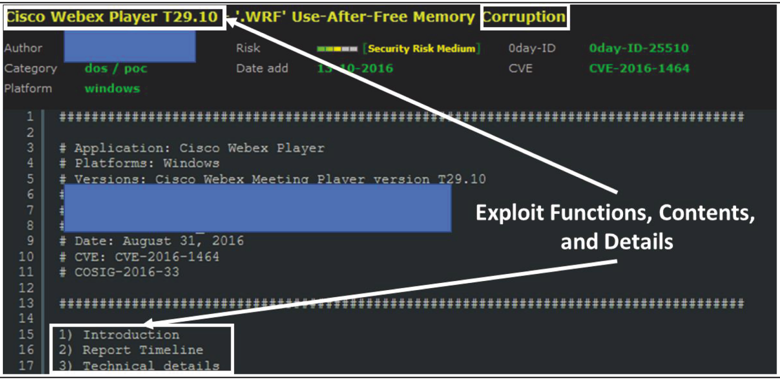
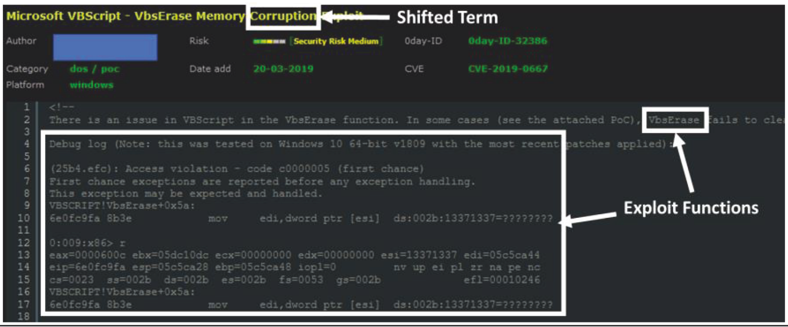
Semantic Shifts of Terms for Web Application Threats					
Rank	Word	Amount Shifted*	Rank	Word	Amount Shifted*
1	sqli	7.447353	11	Filr	5.896086
2	Settings	6.961177	12	vuln	5.877753
3	takeover	6.748133	13	Cuentas	5.872392
4	DISCLOSURE	6.55786	14	Electric	5.862406
5	metacharacters	6.253232	15	bypassing	5.854489
6	Textfilter	6.19481	16	InjecTion	5.834347
7	writeup	6.179463	17	passwords	5.821093
8	filter	6.135306	18	Debit	5.720313
9	IPRC	6.043778	19	Rendicion	5.666406
10	Permission	5.906195	20	Privileges	5.637859
Semantic Shifts of Terms for DoS Threats					
Rank	Word	Amount Shifted*	Rank	Word	Amount Shifted*
1	call	5.676367	11	access	4.822627
2	CSOWNDC	5.612645	12	management	4.753818
3	Corrupt	5.583253	13	OpMinInAnArray	4.609549
4	CSCLASSDC	5.333942	14	EMREXTESCAP	4.551099
5	UserDefined	5.220336	15	Lacking	4.502576
6	console	5.099514	16	Consider	4.467911
7	Explicitly	5.045209	17	sftpget	4.459338
8	Between	4.940183	18	Switch	4.451778
9	Properly	4.920918	19	Traversal	4.428673
10	IsLoopPrePass	4.880469	20	Triggerable	4.42134

and directories, "bypassing" for circumventing security controls, and others. With respect to DoS threats, shifts included "call" for calling functions to execute the attack, "corrupts" for altering the integrity of the specified target, and others. A top word providing actionable intelligence for the DoS is "Corrupt," which appears at rank 3 (shift of 5.583). Such details cannot be attained by conventional temporal graph metrics (e.g., degree).

Table 16 provides a representative sample post from three time points, 2012, 2016, and 2019, to illustrate how the meaning of these terms have shifted. Given that the DoS threats are on more of an uptick than the web applications, and considering space, we present only the shift in DoS. However, interested readers, especially those working within the security industry, can contact the authors to receive additional reports on selected web application exploit evolutions. We also provide the entire dataset such that interested readers can access and explore the data. To protect ourselves from hackers within this community, we anonymize the author and related information.

In its appearances, "corrupt" pertained primarily to attacking memory to deny service to a Windows machine. However, the mechanism that it used to execute the threat varied. In 2012 (top post), "corrupt" pertained to attacking RealPlayer, a program commonly installed on Windows XP machines. At that time, XP was still a prevailing operating system across the globe. In 2016, XP had reached end of life. As a result, the focus of executing memory corruptions shifted to other technologies. In the case of the 2016 post (middle of Table 15), a target program was Cisco Webex Player. This program was and currently is a common video-conferencing software that appears in many commercial and enterprise level networks. By 2019 (bottom post), the focus of corruptions shifted to attacking the memory of Windows 7 systems. A key motivator behind this shift was the end-of-life for Windows 7, which was scheduled to occur on January 14, 2020. Given that this post was made in mid-2019, this discovery can provide tactical intelligence of which systems to

Table 16. Selected DoS Threat Postings with Term "Corrupt"

Year-End	Evolution of DoS Threat Term "Corrupt"
2012	
2016	
2019	

2012:

RealPlayer .mp4 file handling memory corruption

```
Author    [redacted]        Risk       [Security Risk Critical]    0day-ID  0day-ID-17823
Category  dos / poc         Date add   24-03-2012
Platform  windows
```

```
#########################################################################################
Application : RealPlayer mp4 file handling memory corruption
Versions : <= SP 1.1.4 (Build 12.0.0.756)
           <= v15
Date : 21/03/2012
Tested : Windows XP SP3 EN
[redacted]
CVE : 2012-1904
#########################################################################################

Bug :
----
Memory corruption during the handling of the MP4 files, sub_61177240 this function is read
some values from file so this point can be vulnerable because of not checking the values.
>From mp4fformat.dll :

61177240   55               PUSH EBP
61177241   8BEC             MOV EBP,ESP
61177243   8B4D 0C          MOV ECX,DWORD PTR SS:[EBP+C]
61177246   85C9             TEST ECX,ECX                          ; ECX = 00000001
```

2016:

Cisco Webex Player T29.10 - '.WRF' Use-After-Free Memory Corruption

```
Author    [redacted]        Risk       [Security Risk Medium]   0day-ID  0day-ID-25510
Category  dos / poc         Date add   13-10-2016               CVE      CVE-2016-1464
Platform  windows
```

```
##########################################################################

# Application: Cisco Webex Player
# Platforms: Windows
# Versions: Cisco Webex Meeting Player version T29.10
# [redacted]
#
#
# Date: August 31, 2016
# CVE: CVE-2016-1464
# COSIG-2016-33

##########################################################################

1) Introduction
2) Report Timeline
3) Technical details
```

2019:

Microsoft VBScript - VbsErase Memory Corruption Exploit

```
Author    [redacted]        Risk       [Security Risk Medium]   0day-ID  0day-ID-32386
Category  dos / poc         Date add   20-03-2019               CVE      CVE-2019-0667
Platform  windows
```

```
<!--
There is an issue in VBScript in the VbsErase function. In some cases (see the attached PoC), VbsErase fails to cle

Debug log (Note: this was tested on Windows 10 64-bit v1809 with the most recent patches applied):

(25b4.efc): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
VBSCRIPT!VbsErase+0x5a:
6e0fc9fa 8b3e            mov     edi,dword ptr [esi]  ds:002b:13371337=????????

0:009:x86> r
eax=0000600c ebx=05dc10dc ecx=00000000 edx=00000000 esi=13371337 edi=05c5ca44
eip=6e0fc9fa esp=05c5ca28 ebp=05c5ca48 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
VBSCRIPT!VbsErase+0x5a:
6e0fc9fa 8b3e            mov     edi,dword ptr [esi]  ds:002b:13371337=????????
```

proactively remediate. Such a shift cannot be identified by existing methods of identifying emerging threats presented in extant literature as they do not systematically weight and prioritize threat terms in the same fashion as D-GEF.

7.4 Cybersecurity Implications and Potential Future Directions

As discussed in the introduction, CTI is fundamentally a data-driven process that aims to identify emerging threats and key threat actors to enable cybersecurity decision making. Given the broad nature of the discipline, CTI is often segmented into strategic, operational, and tactical levels of focus. Each level has specified job roles and responsibilities as it pertains to the overall CTI lifecycle. Novel CTI computational frameworks, systems, and approaches should ultimately offer value to one or more of these levels.

Table 17. Summary of Selected Security Implications for D-GEF for Varying Levels of Cyber Threat Intelligence (CTI)

Intelligence Level	Selected Stakeholders	Relevant D-GEF Component(s)	Security Implications
Strategic	Security managers, CISO	Time-spell construction	Development of publicly accessible CTI reports; relevant security investments
		Network level statistics	
Operational	IR Team, Security Forensics	Text graph visualizations	Reducing information overload; threat prioritization
		Unsupervised graph embedding generation	Automated and dynamic representation of threat terms
		Top degree statistics	Identification of key threat functions for mitigation; integration into MAEC, STIX
Tactical	Malware Analysts, SOC Analysts		
		Semantic shifts	Identification of IoCs

* *Note*: CISO = Chief Information Security Officer (CISO); SOC = Security Operations Center (SOC); MAEC = Malware Attribute Enumeration and Characterization; STIX = Structured Threat Information Expression; IoC = Indicators of Compromise.

Each component of the D-GEF holds value to selected stakeholders. Table 17 provides a brief summary of the different intelligence levels, examples of selected stakeholders, the relevant D-GEF component(s) that can offer value to those stakeholders, and selected security implications. The following sub-sections provide a further discussion of each level. In each sub-section, we also summarize promising future directions for significantly extending D-GEF to provide further value to each stakeholder group. We note that the value, stakeholders, and implications discussed are not exhaustive, nor are they mutually exclusive, and all require extensive future studies to determine the extent of their validity and proof of self-sustaining use.

7.4.1 Strategic Intelligence. Strategic level CTI focuses on allocating appropriate resources to relevant cybersecurity tasks within a given business unit or organization. Common jobs within this level include CISO's and IT Security Managers. A common mechanism for these roles to perform their job function is the use of quarterly and annual reports (e.g., Verizon Data Breach Report). These colorful and illustrative documents play a crucial role in communicating statistics, visualizations, and other essential content to communicate with executive management and other stakeholders to attain resources and properly invest in cybersecurity mechanisms.

The time-spell construction, network level statistics, and text graph visualization aspects of the D-GEF most directly pertain to prevailing strategic intelligence activities. The time-spell construction combined with the network level statistics supports the quarterly reporting of emerging threat trends reporting of threats. The text graph visualizations can also play a significant role in presenting striking visualizations to guide security investments. For example, through our case studies, we noted a significant increase in the quantities of DoS threats (as per the network level statistics and visualizations). This knowledge can assist in proactively allocating resources to mitigate DoS attacks against Windows systems. Future directions to improve D-GEF for strategic intelligence purposes can be employing deep Bayesian forecasting principles to predict threats at future time-spells with degrees of probability. Such predictions can assist in communicating industry level predictions for future threats.

7.4.2 Operational Intelligence. Operational level CTI is commonly concerned with hunting for threats, understanding their details, remediating them, and communicating their findings to the strategic and tactical levels. Common jobs that relate directly to this level include Incident Response and security forensics teams. These groups often face numerous issues, most notably information overload from the vast quantities of heterogeneous data. As a result, it is often a difficult and non-trivial task for them to prioritize relevant threats.

D-GEF's text graph visualizations, unsupervised graph embedding generation, and top degree statistics offer valuable mechanisms to alleviate some of these issues. The visualizations offer a visual and intuitive approach to bypassing large tables of data and quickly make determinations of subsequent prioritization activities. Unsupervised graph embedding generation provides analysts the ability to dynamically create a representation of threats without any prior knowledge or training datasets. In addition to helping facilitate diachronic analysis, this embedding can offer significant value in categorizing and clustering threats and target platforms. Finally, the top degree statistics can help prioritize top threats and facilitate subsequent remediation activities.

Future work to significantly extend D-GEF to offer additional operational intelligence include creating linkages between discovered threats and their vulnerabilities to develop comprehensive cyber-defenses, report key hackers behind emerging threats to help support attribution, and fusing traditional social media data sources (e.g., Twitter, news articles) or hacker community platform data (e.g., DarkNet marketplaces) with forum data to identify how threats propagate through cyberspace. Additional examination of how these emerging threats relate to vulnerabilities in varying IoT devices described in past literature can also yield holistic CTI [13, 20, 28, 39, 44, 45, 56]. The approach can also have value in contexts beyond cybersecurity, including emerging health analytics and IoT applications [65, 66].

7.4.3 Tactical Intelligence. Tactical intelligence focuses on monitoring, escalating, and detecting Indicators of Compromise (IoCs), executing remediation exercises (e.g., patching vulnerable systems), and feeding the indicators into selected security systems and reporting formats. Common job roles include SOC analysts and malware analysts. Common challenges that these job functions face include threat identification and prioritization [3, 40]. D-GEF's computation of top degree statistics and semantic shifts can offer significant value to help address these issues. Both provide valuable tactical leads for CTI professionals. One use case for the generated intelligence would be the integration of new rules into Security Information and Event Management (SIEM) systems (e.g., Splunk). SIEMs are used by many CTI professionals to monitor the status of machines on a network and detect IoCs. Using the intelligence provided by the identification of semantic shifts, the SIEM would monitor if the listed terms appear as file names or other objects within their network. Should they exist, the SIEM can quarantine the machine and generate alerts to systems administrators who can mitigate the threat. Future D-GEF extensions can monitor these semantic shifts over time and implement novel diachronic anomaly detection to automatically present prioritized threats based on unusually high (i.e., spiky) semantic shifts. Additional promising avenues can be automatically inputting the detected threat terms into larger cybersecurity frameworks and reporting, such as STIX, MAEC, and MITRE ATT&CK.

8 CONCLUSIONS AND FUTURE DIRECTIONS

Preventing cyber-attacks has become a grand societal challenge. CTI has emerged as a viable approach to combat this issue. However, existing CTI practices are reactive to threats already used in cyber-attacks. Consequently, breaches are on an unfortunate and dangerous increase. CTI professionals have pointed to the online hacker community as a novel data source to proactively identify emerging threats. Among other hacker community platforms, forums allow hackers to freely share a large scale of malicious threats. Despite its CTI promise, hacker forums possess unique data characteristics which necessitate that novel, customized CTI analytics.

In this article, we examine online hacker forums to identify emerging threats and trends. To achieve this objective, we developed a novel D-GEF (Diachronic Graph Embedding Framework).

This innovative framework makes several key research contributions. First, it operates on a GoW (Graph-of-Words) representation of hacker forum threat text to create low-dimensional word embeddings in an unsupervised fashion. Moreover, its unsupervised nature allows it to work on any dataset size, ideal for contexts that lack sufficient training data (e.g., hacker community analysis). Third, semantic displacement measures adopted from diachronic linguistics literature map the evolution of hacker terminology over multiple time-spells. Finally, a series of benchmark evaluations reveals D-GEF's superior performance over prevailing word and graph embedding approaches in selected downstream tasks. To the best of our knowledge, this is the first study that employs graph embeddings in lieu of traditional word embedding analysis for a diachronic linguistics CTI task.

D-GEF's practical utility is demonstrated with an in-depth case study of ransomware in a long-standing English hacker forum. By constructing text graphs for multiple time-spells, we identified the overall trends of when and how selected web applications and DoS threats were posted. More importantly, we pinpointed how specific words shifted in their meaning. Identifying these semantic displacements helped detect emerging ransomware functionalities. Each discovery can provide valuable, actionable CTI for selected professionals to proactively deploy appropriate security controls. While demonstrated in hacker forums, the D-GEF can be leveraged by scholars for other novel, high-impact cybersecurity applications. Selected examples or recent interests include enhancing memory forensics [33] and malware evolution on datasets extracted from VirusTotal [53]. Each direction can develop proactive CTI capabilities to ultimately create a safer and more secure society.

ACKNOWLEDGMENTS

We would like to thank the three anonymous reviewers, associate editor, and editor-in-chief for their insightful comments. Their feedback helped greatly improve this work.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on the World Wide Web (2013) (WWW 2013)*, 37–47. DOI: https://doi.org/10.1145/2488388.2488393

[2] Camilo Akimushkin, Diego Raphael Amancio, and Osvaldo Novais Oliveira. 2017. Text authorship identified using the dynamics of word co-occurrence networks. *PLoS One* 12, 1 (2017). DOI: https://doi.org/10.1371/journal.pone.0170527

[3] Nolan Arnold, Mohammadreza Ebrahimi, Ning Zhang, Ben Lazarine, Mark Patton, Hsinchun Chen, and Sagar Samtani. 2019. Dark-net ecosystem cyber-threat intelligence (CTI) tool. In *Proceedings of the 2019 IEEE International Conference on Intelligence and Security Informatics (ISI 2019)*. DOI: https://doi.org/10.1109/ISI.2019.8823501

[4] Amir Bakarov. 2018. A survey of word embeddings evaluation methods. (January 2018). Retrieved from http://arxiv.org/abs/1801.09536

[5] Albert-László Barabási. 2016. *Network Science* (1st ed.). Cambridge University Press.

[6] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* 15, (2003), 1–24. Retrieved from papers2://publication/uuid/DF584584-4B36-4A91-B355-2EB36C58277B.

[7] Victor Benjamin and Hsinchun Chen. 2015. Developing understanding of hacker language through the use of lexical semantics. In *Proceedings of the 2015 IEEE International Conference on Intelligence and Security Informatics: Securing the World through an Alignment of Technology, Intelligence, Humans and Organizations (ISI 2015)*, 79–84. DOI: https://doi.org/10.1109/ISI.2015.7165943

[8] Victor Benjamin, Weifeng Li, Thomas Holt, and Hsinchun Chen. 2015. Exploring threats and vulnerabilities in hacker web: Forums, IRC and carding shops. In *Proceedings of the 2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 85–90. DOI: https://doi.org/10.1109/ISI.2015.7165944

[9] Matt Bromiley. 2016. Threat Intelligence: What it is, and how to use it effectively. *SANS Institute*. Retrieved June 5, 2017 from https://www.sans.org/reading-room/whitepapers/analyst/threat-intelligence-is-effectively-37282.

[10] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15)*, 891–900. DOI: https://doi.org/10.1145/2806416.2806512

[11] Po-Yi Du, Ning Zhang, Mohammedreza Ebrahimi, Sagar Samtani, Ben Lazarine, Nolan Arnold, Rachael Dunn, Sandeep Suntwal, Guadalupe Angeles, Robert Schweitzer, and Hsinchun Chen. 2018. Identifying, collecting, and presenting hacker community data: Forums, IRC, carding shops, and DNMs. In *Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 70–75. DOI: https://doi.org/10.1109/ISI.2018.8587327

[12] Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Rebecca S. Portnoff, Sadia Afroz, Damon McCoy, Kirill Levchenko, and Vern Paxson. 2017. Identifying products in online cybercrime marketplaces: A dataset for fine-grained domain adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*. DOI: https://doi.org/10.18653/v1/d17-1275

[13] Malaka El, Emma McMahon, Sagar Samtani, Mark Patton, and Hsinchun Chen. 2017. Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments. In *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 83–88. DOI: https://doi.org/10.1109/ISI.2017.8004879

[14] Katheryn A. Farris, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. VULCON: A system for vulnerability prioritization, mitigation, and management. *ACM Trans. Priv. Secur.* 21, 4 (June 2018), 1–28. DOI: https://doi.org/10.1145/3196884

[15] Palash Goyal, K. S. M. Tozammel Hossain, Ashok Deb, Nazgol Tavabi, Nathan Bartley, Andr'es Abeliuk, Emilio Ferrara, and Kristina Lerman. 2018. Discovering signals from Web sources to predict cyber attacks. (June 2018). Retrieved from http://arxiv.org/abs/1806.03342

[16] John Grisham, Sagar Samtani, Mark Patton, and Hsinchun Chen. 2017. Identifying mobile malware and key threat actors in online hacker forums for proactive cyber threat intelligence. In *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 13–18. DOI: https://doi.org/10.1109/ISI.2017.8004867

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, 855–864. DOI: https://doi.org/10.1145/2939672.2939754

[18] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. (May 2016). Retrieved from http://arxiv.org/abs/1605.09096

[19] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods of Natural Language Process* (November 2016), 2116–2121. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/28580459.

[20] Christopher R. Harrell, Mark Patton, Hsinchun Chen, and Sagar Samtani. 2018. Vulnerability assessment, remediation, and automated reporting: Case studies of higher education institutions. In *Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI 2018)*. DOI: https://doi.org/10.1109/ISI.2018.8587380

[21] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A *k*-means clustering algorithm. *Appl. Stat.* 28, 1 (1979), 100. DOI: https://doi.org/10.2307/2346830

[22] Yuexian Hou, Peng Zhang, Xingxing Xu, Xiaowei Zhang, and Wenjie Li. 2009. Nonlinear dimensionality reduction by locally linear inlaying. *IEEE Trans. Neural Networks* 20, 2 (2009), 300–315. DOI: https://doi.org/10.1109/TNN.2008.2005582

[23] Alice Hutchings and Thomas J. Holt. 2015. A crime script analysis of the online stolen data market. *Br. J. Criminol.* 55, 3 (May 2015), 596–614. DOI: https://doi.org/10.1093/bjc/azu106

[24] Rob Jansen, Matthew Traudt, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. 2018. KIST: Kernel-Informed Socket Transport for Tor. *ACM Trans. Priv. Secur.* 22, 1 (December 2018), 1–37. DOI: https://doi.org/10.1145/3278121

[25] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop (NIPS 2016)*. Retrieved from http://arxiv.org/abs/1611.07308

[26] Rob Lee and Robert M. Lee. 2017. The hunter strikes back: The 2017 threat hunting survey. SANS Institute. Retrieved January 11, 2018 from https://www.sans.org/reading-room/whitepapers/analyst/hunter-strikes-back-2017-threat-hunting-survey-37760.

[27] Weifeng Li, Junming Yin, and Hsinchun Chen. 2016. Targeting key data breach services in underground supply chain. In *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 322–324. DOI: https://doi.org/10.1109/ISI.2016.7745501

[28] Emma McMahon, Mark Patton, Sagar Samtani, and Hsinchun Chen. 2018. Benchmarking vulnerability assessment tools for enhanced cyber-physical system (CPS) Resiliency. In *Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018*. DOI: https://doi.org/10.1109/ISI.2018.8587353

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NPIS'13)*, 3111–3119. Retrieved from http://arxiv.org/abs/1310.4546

[30] Vivi Nastase, Rada Mihalcea, and Dragomir R. Radev. 2015. A survey of graphs in natural language processing. *Nat. Lang. Eng.* 21, 05 (November 2015), 665–698. DOI: https://doi.org/10.1017/S1351324915000340

[31] Eric Nunes, Ahmad Diab, Andrew Gunn, Ericsson Marin, Vineet Mishra, Vivin Paliath, John Robertson, Jana Shakarian, Amanda Thart, and Paulo Shakarian. 2016. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 7–12. DOI: https://doi.org/10.1109/ISI.2016.7745435

[32] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the ACM SIGKDD International Conference on Knoelwdge Discovery Data Mining* (Aug. 2016), 1105–1114. DOI: https://doi.org/10.1145/2939672.2939751

[33] Fabio Pagani, Oleksii Fedorov, and Davide Balzarotti. 2019. Introducing the temporal dimension to memory forensics. *ACM Trans. Priv. Secur.* 22, 2 (Mar. 2019), 1–21. DOI: https://doi.org/10.1145/3310355

[34] Sergio Pastrana and Guillermo Suarez-Tangil. 2019. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. (January 2019). Retrieved from http://arxiv.org/abs/1901.00846

[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*, 701–710. DOI: https://doi.org/10.1145/2623330.2623732

[36] Giulio Ermanno Pibiri and Rossano Venturini. 2019. Handling massive *n*-gram datasets efficiently. *ACM Trans. Inf. Syst.* 37, 2 (Feb. 2019), 1–41. DOI: https://doi.org/10.1145/3302913

[37] Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 410–420.

[38] Francois Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association of Computer Linguistics and the 7th International Joint Conference on the Natural Language Processes (Volume 1 Long Papers)* (2015), 1702–1712. Retrieved from http://www.aclweb.org/anthology/P15-1164.

[39] Morteza Safaei Pour, Antonio Mangino, Kurt Friday, Matthias Rathbun, Elias Bou-Harb, Farkhund Iqbal, Sagar Samtani, Jorge Crichigno, and Nasir Ghani. 2020. On data-driven curation, learning, and analysis for inferring evolving internet-of-Things (IoT) botnets in the wild. *Comput. Secur.* (2020). DOI: https://doi.org/10.1016/j.cose.2019.101707

[40] Sagar Samtani, Maggie Abate, Victor Benjamin, and Weifeng Li. 2020. Cybersecurity as an industry: A cyber threat intelligence perspective. In *The Palgrave Handbook of International Cybercrime and Cyberdeviance*. DOI: https://doi.org/10.1007/978-3-319-78440-3_8

[41] Sagar Samtani, Kory Chinn, Cathy Larson, and Hsinchun Chen. 2016. A secure hacker assets portal: Cyber threat intelligence and malware analysis. In *Proceedings of the 2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 19–24. DOI: https://doi.org/10.1109/ISI.2016.7745437

[42] Sagar Samtani, Ryan Chinn, and Hsinchun Chen. 2015. Exploring hacker assets in underground forums. In *Proceedings of the 2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 31–36. DOI: https://doi.org/10.1109/ISI.2015.7165935

[43] Sagar Samtani, Ryan Chinn, Hsinchun Chen, and Jay F. Nunamaker. 2017. Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence. *J. Manag. Inf. Syst.* 34, 4 (2017), 1023–1053.

[44] Sagar Samtani, Shuo Yu, Hongyi Zhu, Mark Patton, and Hsinchun Chen. 2016. Identifying SCADA vulnerabilities using passive and active vulnerability assessment techniques. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data (ISI 2016)*. DOI: https://doi.org/10.1109/ISI.2016.7745438

[45] Sagar Samtani, Shuo Yu, Hongyi Zhu, Mark Patton, John Matherly, and HsinChun Chen. 2018. Identifying SCADA systems and their vulnerabilities on the Internet of Things: A text-mining approach. *IEEE Intell. Syst.* 33, 2 (March 2018), 63–73. DOI: https://doi.org/10.1109/MIS.2018.111145022

[46] Anna Sapienza, Alessandro Bessi, Saranya Damodaran, Paulo Shakarian, Kristina Lerman, and Emilio Ferrara. 2017. Early warnings of cyber threats in online discussions. In *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 667–674. DOI: https://doi.org/10.1109/ICDMW.2017.94

[47] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 298–307. DOI: https://doi.org/10.18653/v1/D15-1036

[48] Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31, 1 (Mar. 1966), 1–10. DOI: https://doi.org/10.1007/BF02289451

[49] Dave Shackleford. 2016. *2016 Security Analytics Survey*. SANS Institute. Retrieved June 5, 2017 from https://www.sans.org/reading-room/whitepapers/analyst/2016-security-analytics-survey-37467.

[50] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, Min Lyu, and Hongxia Jin. 2017. Differentially Private *k*-means clustering and a hybrid approach to private optimization. *ACM Trans. Priv. Secur.* 20, 4 (Oct. 2017), 1–33. DOI: https://doi.org/10.1145/3133201

[51] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15)*, 1067–1077. DOI: https://doi.org/10.1145/2736277.2741093

[52] Nazgol Tavabi, Palash Goyal, Mohammed Almukaynizi, Paulo Shakarian, and Kristina Lerman. 2018. Font Size: Dark-Embed: Exploit prediction with neural language models. In *Proceedings of the 30th AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*, 7849–7854.

[53] Xabier Ugarte-Pedrero, Mariano Graziano, and Davide Balzarotti. 2019. A close look at a daily dataset of malware samples. *ACM Trans. Priv. Secur.* 22, 1 (Jan. 2019), 1–30. DOI: https://doi.org/10.1145/3291061

[54] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C. C. Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *APSIPA Trans. Sig. Inf. Proc.* DOI: https://doi.org/10.1017/ATSIP.2019.12

[55] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2016), 1225–1234. DOI: https://doi.org/10.1145/2939672.2939753

[56] Ryan Williams, Emma McMahon, Sagar Samtani, Mark Patton, and Hsinchun Chen. 2017. Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach. In *Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data (ISI 2017).* DOI: https://doi.org/10.1109/ISI.2017.8004904

[57] Ryan Williams, Sagar Samtani, Mark Patton, and Hsinchun Chen. 2018. Incremental hacker forum exploit collection and classification for proactive cyber threat intelligence: An exploratory study. In *Proceedings of the 2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 94–99. DOI: https://doi.org/10.1109/ISI.2018.8587336

[58] Hao Yang, Xiulin Ma, Kun Du, Zhou Li, Haixin Duan, Xiaodong Su, Guang Liu, Zhifeng Geng, and Jianping Wu. 2017. How to learn Klingon without a dictionary: Detection and measurement of black keywords used by the underground economy. In *Proceedings of the IEEE Symposium on Security and Privacy.* DOI: https://doi.org/10.1109/SP.2017.11

[59] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM'18)*, 673–681. DOI: https://doi.org/10.1145/3159652.3159703

[60] Kan Yuan, Haoran Lu, Xiaojing Liao, and Xiao Feng Wang. 2018. Reading thieves' cant: Automatically identifying and understanding dark jargons from cybercrime marketplaces. In *Proceedings of the 27th USENIX Security Symposium.*

[61] Michael Zhai, Johnny Tan, and Jinho D. Choi. 2016. Intrinsic and extrinsic evaluations of word embeddings. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (2016), 4282–4283. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12454/12257.

[62] Yiming Zhang, Yujie Fan, Shifu Hou, Jian Liu, Yanfang Ye, and Thirimachos Bourlai. 2018. Idetector: Automate underground forum analysis based on heterogeneous information network. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 1071–1078. DOI: https://doi.org/10.1109/ASONAM.2018.8508414

[63] Kangzhi Zhao, Yong Zhang, Chunxiao Xing, Weifeng Li, and Hsinchun Chen. 2016. Chinese underground market jargon analysis based on unsupervised learning. In *IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016.* DOI: https://doi.org/10.1109/ISI.2016.7745450

[64] Ziming Zhao, Mukund Sankaran, Gail-Joon Ahn, Thomas J. Holt, Yiming Jing, and Hongxin Hu. 2016. Mules, seals, and attacking tools: Analyzing 12 online marketplaces. *IEEE Secur. Priv.* 14, 3 (May 2016), 32–43. DOI: https://doi.org/10.1109/MSP.2016.46

[65] Hongyi Zhu, Sagar Samtani, Randall Brown, and Hsinchun Chen. 2020. A deep learning approach for recognizing activity of daily living (ADL) for senior care: Exploiting interaction dependency and temporal patterns. *Forthcom. MIS Q.* (2020). Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3595738.

[66] Hongyi Zhu, Sagar Samtani, Hsinchun Chen, and Jay F Nunamaker, Jr. 2020. Human identification for activities of daily living: A deep transfer learning approach. *J. Manag. Inf. Syst.* 37, 2 (2020), 457–483. DOI: https://doi.org/10.1080/07421222.2020.1759961

Received March 2020; revised March 2020; accepted July 2020