

DEVELOPING PROACTIVE CYBER THREAT INTELLIGENCE FROM THE ONLINE
HACKER COMMUNITY: A COMPUTATIONAL DESIGN SCIENCE APPROACH

by

Sagar Samtani

Copyright © Sagar Samtani 2018

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2018

ProQuest Number: 10830193

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10830193

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

As members of the Dissertation Committee, I certify that I have read the dissertation prepared by Sagar Samtani, titled Developing Proactive Cyber Threat Intelligence from the Online Hacker Community: A Computational Design Science Approach and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.



Hsinchun Chen Date: 03/30/2018



Jay F. Nunamaker, Jr. Date: 03/30/2018



Susan A. Brown Date: 03/30/2018

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.



Dissertation Director: Hsinchun Chen Date: 03/30/2018

ARIZONA

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of the requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, However, permission must be obtained from the author.

SIGNED: Sagar Samtani

ACKNOWLEDGEMENTS

I hold great appreciation for my dissertation committee members, Drs. Hsinchun Chen, Jay F. Nunamaker, Jr., and Susan A. Brown. Each taught me critical skills that I will carry throughout my academic career. I am especially grateful to my advisor Dr. Chen. By working closely with him on many meaningful projects, he instilled in me critical life lessons about respect, work ethic, balance, attitude, teamwork, vision, and most importantly, engaging in activities that have a positive societal impact.

I thank my Ph.D., MS, and BS colleagues in the Artificial Intelligence Lab. Special thanks to Cathy Larson for her optimism, patience, friendship, and well-placed humorous comments, especially in the formative years of my Ph.D. studies. Many thanks to Hongyi Zhu for his support during the final months of my Ph.D. studies. I thank staff members Riley McIsaac, Resha Shenandoah, and Andy Pressman for their support in various lab activities. I would also like to express my gratitude to Dr. Mark Patton for providing guidance and feedback during my graduate studies. I am also very grateful to my friends and family, particularly my parents, for supporting me through this phase of life.

I thank the National Science Foundation's (NSF's) Scholarship-for-Service (SFS) (DUE-1303362), Secure and Trustworthy Cyberspace (SaTC) (SES-1314631), and Data Infrastructure Building Blocks (DIBBs) (ACI-1443019) programs for providing funding for my work. I also thank collaborators at National Cyber Forensics Training Alliance (NCFTA), Policing in Cyberspace (POLCYB), and John Matherly of Shodan. All of you helped me think carefully about the practical impact and value of my research.

DEDICATION

This dissertation is dedicated to my late grandfather Chandan R. Samtani, a man who had a profound impact on me in my youth.

TABLE OF CONTENTS

LIST OF FIGURES	9
LIST OF TABLES	10
ABSTRACT	12
1. INTRODUCTION	13
2. ESSAY I: EXPLORING HACKER ASSETS IN ONLINE HACKER FORUMS	22
2.1. Introduction	22
2.2. Literature Review	24
2.2.1. Hacker Community Research	24
2.2.2. Cyber Threat Intelligence (CTI)	28
2.2.3. Source Code Analysis	29
2.2.4. Social Network Analysis (SNA)	32
2.3. Research Gaps and Questions	33
2.4. Research Design and Testbed	34
2.4.1. Data Collection and Pre-Processing	35
2.4.2. Asset Analysis and Evaluations	38
2.4.3. Social Network Construction	40
2.5. Results and Discussion	41
2.5.1. Classification Performance	41
2.5.2. LDA Evaluations	42
2.5.3. Malicious and Emerging Source Code Topics	43
2.5.4. Malicious and Emerging Attachment Topics	49
2.5.5. Malicious and Emerging Tutorial Topics	51
2.5.6. AZSecure Hacker Assets Portal	53
2.5.7. Social Network Analysis	55
2.6. Conclusion and Future Directions	58
3. ESSAY II: LINKING HACKER COMMUNITY EXPLOITS TO KNOWN VULNERABILITIES: A DEEP STRUCTURED SEMANTIC MODEL APPROACH	60
3.1. Introduction	60
3.2. Literature Review	63
3.2.1. Hacker Community Research	63
3.2.2. Vulnerability Assessment	64
3.2.3. Deep Structured Semantic Model (DSSM)	66
3.3. Research Gaps and Questions	72
3.4. Research Design and Testbed	73

3.4.1. Data Collection: Hacker Forum Tool Extraction and Vulnerability Assessment Data	73
3.4.2. Exploit Vulnerability Linkages: Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM)	74
3.4.3. Exploit-Vulnerability Linkages: Benchmark Evaluations	78
3.4.4. Exploit-Vulnerability Linkages: Device Vulnerability Severity Metric (DVSM) Calculation	79
3.4.5. Case Studies: US Hospitals and SCADA Devices	81
3.5. Results and Discussion	83
3.5.1. EV-DSSM Baseline Evaluation	83
3.5.2. EV-DSSM Word Hashing Evaluations	86
3.5.3. US Hospital Case Study	88
3.5.4. SCADA Device Case Study	90
3.6. Conclusion and Future Directions	92
4. ESSAY III: IDENTIFYING EXPLOIT SHARING HACKERS AND COMMUNITIES: A GRAPH CONVOLUTIONAL AUTOENCODER APPROACH	95
4.1. Introduction	95
4.2. Literature Review	98
4.2.1. Hacker Community Research	98
4.2.2. Social Network Analysis (SNA): Centrality Measures and Community Detection	102
4.2.3. Graph Convolutional Networks (GCN)	104
4.2.4. Autoencoders	107
4.3. Research Gaps and Questions	108
4.4. Research Design and Testbed	109
4.4.1. Data Collection and Pre-Processing	110
4.4.2. Network Representation: Formulation, Descriptive Statistics, and Exploit Degree Centrality (EDC)	111
4.4.3. Graph Convolutional Autoencoder (GCAE)	112
4.4.4. GCAE Evaluations	116
4.4.5. Key Hacker Identification and Community Detection: Case Study	119
4.5. Results and Discussion: Experiments and Case Study	120
4.5.1. Experiment 1: GCAE vs Structural Embeddings	120
4.5.2. Experiment 2: GCAE vs Sparse and Denoising GCAE	122
4.5.3. Case Study Results	123
4.6. Conclusion and Future Directions	129

5. ESSAY IV: IDENTIFYING EMERGING HACKER EXPLOITS: A DIACHRONIC GRAPH CONVOLUTIONAL AUTOENCODER FRAMEWORK	131
5.1. Introduction.....	131
5.2. Literature Review.....	134
5.2.1. Hacker Community Research	134
5.2.2. Text Graphs.....	137
5.2.3. Graph Convolutional Networks (GCNs)	138
5.2.4. Autoencoders	140
5.2.5. Diachronic Word Embeddings.....	142
5.3. Research Gaps and Questions	144
5.4. Research Design and Testbed	145
5.4.1. Data Collection and Pre-Processing.....	145
5.4.2. Time-Spell and Text Graph Construction.....	146
5.4.3. Diachronic Graph Convolutional Autoencoder (D-GCAE) Framework Development	147
5.4.4. D-GCAE Evaluations.....	152
5.4.5. Exploit Trend Identification: Case Study	157
5.5. Results and Discussion	157
5.5.1. Experiment 1: GCAE vs Word2vec.....	157
5.5.2. Experiment 2: GCAE vs Text Graph Embeddings	159
5.5.3. Experiment 3: GCAE vs Sparse and Denoising GCAE.....	160
5.5.4. Ransomware Case Study Results.....	162
5.6. Conclusion and Future Directions	169
6. CONCLUSION AND FUTURE DIRECTIONS.....	172
6.1. Contributions: Practical and to IS Knowledge Base.....	172
6.2. Future Research Directions.....	177
7. REFERENCES	180

LIST OF FIGURES

Figure 1. Four Phase CTI Lifecycle.....	13
Figure 2. Forum Post with BlackPOS Malware.....	15
Figure 3. How to Access Exploits.....	25
Figure 4. Methodological Research Framework for Analyzing Hacker Assets	35
Figure 5. Emerging Malicious Source Code Topics for OpenSC.....	45
Figure 6. Emerging Malicious Source Code Topics in ExploitIN.....	47
Figure 7. Emerging Malicious Source Code Topics for Reverse4You	48
Figure 8. Emerging Malicious Attachment Topics.....	50
Figure 9. Emerging Malicious Tutorial Topics.....	52
Figure 10. AZSecure Hacker Assets Portal	53
Figure 11. AZSecure Hacker Assets Portal Dashboard.....	54
Figure 12. Crypter Bipartite and Monopartite Networks.....	55
Figure 13. Example Forum Posts with Attached Android Exploits.....	61
Figure 14. Sample Vulnerability Descriptions.....	65
Figure 15. Deep Structured Semantic Model (DSSM) Architecture (Adapted from Huang et al. 2013)	69
Figure 16. Illustration of the NDCG Metric	72
Figure 17. Exploit-Vulnerability-Deep Structured Semantic Model (EV-DSSM) Architecture	76
Figure 18. DSSM Benchmark Evaluation Results.....	84
Figure 19. EV-DSSM Performance Example.....	85
Figure 20. DSSM Word Hashing Evaluation Results.....	87
Figure 21. Selected Vulnerabilities from Partners eCare Portal on the 17.x.x.x Hospital Network.....	90
Figure 22. Example of Exploit Sharing in Hacker Forum	96
Figure 23. Autoencoder Architecture	107
Figure 24. Research Framework for Identifying Key Exploit Sharing Hackers and Communities.....	109
Figure 25. Architecture of the Graph Convolutional Autoencoder (GCAE).....	113
Figure 26. GCAE Benchmark Evaluation Results.....	121
Figure 27. Communities Identified Using GCAE Embeddings.....	127
Figure 28. Example of a hacker providing a Bitcoin Miner 0-day exploit.....	132
Figure 29. Autoencoder Operations.....	141
Figure 30. Research Framework for Identifying Emerging Hacker Exploit Trends	145
Figure 31. Architecture of the Graph Convolutional Autoencoder (GCAE).....	148
Figure 32. D-Graph Convolutional Autoencoder (D-GCAE) Procedure	150
Figure 33. GCAE vs Word2Vec Evaluation Results	158
Figure 34. GCAE Benchmark Evaluation Results.....	159
Figure 35. GCAE Architecture Evaluation Results	161
Figure 36. Overall Ransomware Trend (2010 – 2017).....	163
Figure 37. Three representative sample ransomware posts	168

LIST OF TABLES

Table 1. Summary of Selected IS Cybersecurity Literature (Adapted from Hui et al. 2016)	16
Table 2. Genres of Design Science Research	18
Table 3. Overview of Four Essays	19
Table 4. Summary of Research Testbed	37
Table 5. Classification Results	42
Table 6. p-Values for Pair-wise t-tests for SVM against benchmark classifiers	42
Table 7. Optimal Topic Numbers	43
Table 8. Malicious Source Code Topics in OpenSC Forum	44
Table 9. Malicious Source Code Topics in ExploitIN Forum	46
Table 10. Malicious Source Code Topics in Reverse4You Forum	48
Table 11. Malicious Attachment Topics	49
Table 12. Malicious Tutorial Topics	51
Table 13. Topological and Node Level Metrics for Crypter Bipartite and Monopartite Hacker Networks	56
Table 14. Ranking and Forum Status for Key Hackers based on Degree and Betweenness Centrality	57
Table 15. CVSS Score Ranges, Rankings, and Examples	66
Table 16. Selected Literature Using DSSM's	71
Table 17. Summary of Hacker Forum Tool Collection	73
Table 18. Summary of Security Focus Vulnerability Information Collection	74
Table 19. Summary of Benchmark EV-DSSM Evaluations	78
Table 20. Features for Device Vulnerability Severity Metric	80
Table 21. EV-DSSM Benchmark Results	84
Table 22. Word Hashing Evaluation Results	86
Table 23. Selected Hospital Vulnerabilities Their Most Relevant Exploit Identified by the EV-DSSM	88
Table 24. Most Susceptible Device on Each Hospital's Network	89
Table 25. Selected SCADA Vulnerabilities Their Most Relevant Exploit as Identified by the EV-DSSM	91
Table 26. Selected Hacker Forum Literature Identifying Key Hackers and/or Communities	100
Table 27. Standard Centrality Measures Used to Identify Key Nodes in a Network	102
Table 28. Selected Studies Using Graph Convolutional Networks	106
Table 29. Summary of Hacker Forum Collection	110
Table 30. Summary of Benchmark GCAE Experiments	117
Table 31. GCAE Benchmark Evaluation Results	120
Table 32. GCAE Architecture Evaluation Results	122
Table 33. Topological and Node Level Descriptive Statistics	124
Table 34. Ranking of Top Hackers in Constructed Network	125
Table 35. Each Community's Top Hackers (Ranked by EDC)	128
Table 36. Selected Studies Identifying Exploits in Online Hacker Forums	135
Table 37. Selected Studies Using Graph Convolutional Networks on Text Graphs	139
Table 38. Summary of Hacker Forum Collection	146
Table 39. Summary of Benchmark GCAE Experiments	155
Table 40. GCAE vs Word2Vec Evaluation Results	157

Table 41. GCAE Benchmark Evaluation Results	159
Table 42. GCAE Architecture Evaluation Results	160
Table 43. Number of Ransomware Per Quarter (2010 – 2017)	162
Table 44. Topological and Node Level Descriptive Statistics Between 2010 – 2017	164
Table 45. Topological and Node Level Descriptive Statistics Between 2014 – 2017	166
Table 46. Top Shifted Words Between 2010 – 2017	167
Table 47. Each Dissertation Essay’s Contributions to Practice and the IS Knowledge Base	174

ABSTRACT

The proliferation of information systems (IS) has afforded modern society with unprecedented benefits. Unfortunately, malicious hackers often exploit these technologies for cyberwarfare, hacktivism, espionage, or financial purposes, costing the global economy over \$450 billion annually. To combat this issue, many organizations develop Cyber Threat Intelligence, or knowledge about key hackers and emerging threats. Despite CTI's value experts note that existing approaches are reactive in nature. Thus, cyber-attacks remain on an unfortunate uptick. CTI experts have suggested studying the international and ever-evolving online hacker community to address these concerns. However, online hacker community platforms, specifically forums, contain tens of thousands of unstructured, un-sanitized text records. Existing CTI analytics and behavioral and economic methodologies employed in extant IS cybersecurity inquiries were not designed for such data characteristics.

This dissertation presents four essays that adopt the design science approach to develop a series of novel CTI computational IT artifacts to solve a salient CTI issues. Essay I sets the foundation by developing a novel data and text mining framework to automatically extract and categorize malicious hacker exploits. Essay II expands upon this by automatically linking exploits and vulnerabilities detected by modern vulnerability scanners with a novel algorithm, the Exploit-Vulnerability Deep Structured Semantic Model. Essay III leverages graph convolutional networks and autoencoders to develop a novel deep learning architecture to identify the hackers and communities. Essay IV extends the model presented in essay III to identify emerging hacker exploits. Beyond the practical contributions provided by the presented IT artifacts, this dissertation offers numerous design principles to guide future computational cybersecurity IS research and other analytics related research inquiries.

1. INTRODUCTION

The proliferation of computing technologies has afforded modern society with unprecedented benefits. Industry, government, and academia use databases, communication networks, and other information systems (IS) to execute day-to-day operations. Unfortunately, malicious hackers often exploit these systems for cyberwarfare, hacktivism, espionage, or financial purposes, costing the global economy over \$450 billion annually (Graham 2017). To combat this dire issue, many organizations create, manage, and use knowledge about key hackers and emerging threats. This process, also referred to as Cyber Threat Intelligence (CTI), has emerged as a critical aspect of cybersecurity (Shackleford 2017). Figure 1 illustrates the four phases of the CTI lifecycle.

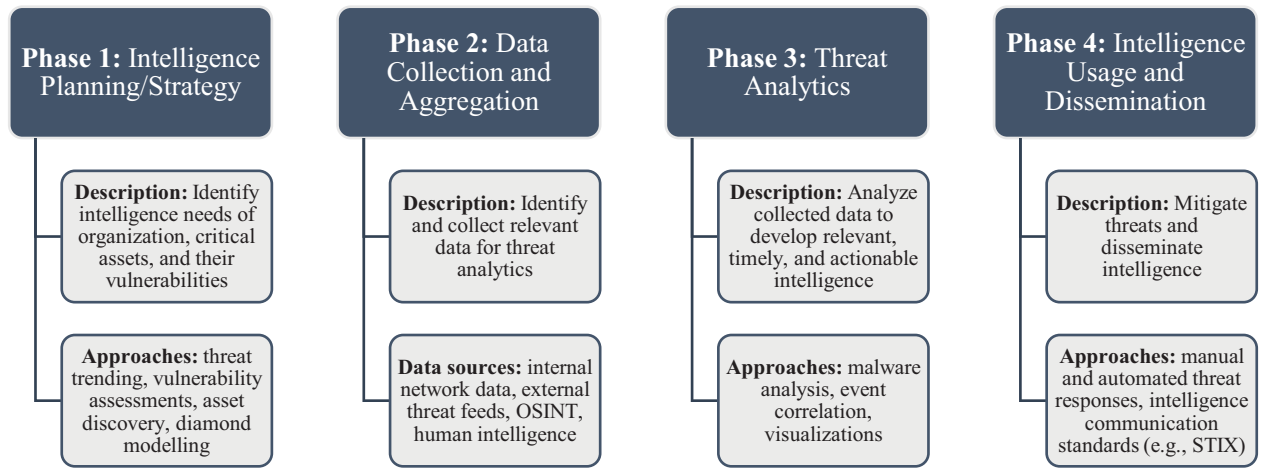


Figure 1. Four Phase CTI Lifecycle

CTI is fundamentally a data-driven process. Organizations will first define their intelligence needs by examining the existing threat landscape, monitoring their cyber-assets, and modelling possible attack vectors. This information guides data collection from Intrusion Detection and Prevention Systems (IDS/IPS), and log files from databases, firewalls, and servers. Well-refined analytics such as malware analysis, event correlation, and forensics to

derive the intelligence needed for CTI professionals to create cyber-defenses (Friedman 2015; Kime 2016; Shackleford 2016). Despite the maturity of these procedures, experts note the reliance on past events (e.g., log files) creates reactive intelligence (Bromiley, 2016). Consequently, major industry firms such as Ernst & Young (EY), have expressed that “cyber threats are increasing,” “businesses still aren’t doing enough to combat them,” and that “organizations need to take a more proactive approach to cybersecurity” (EY, 2014). The SANS institute has further urged the use of “external threat intelligence sources to help alert the organization of threats it was not previously aware of” (Bromiley, 2016).

One data source strongly recommended by CTI experts to generate proactive CTI is the vast and evolving international online hacker community (Bromiley, 2016; Shackleford, 2016). The online hacker community is an appealing and novel CTI data source as it attracts and motivates millions of hackers from the US, Russia, Middle East, and China to share new hacking tools and knowledge. Today, the online hacker community comprises of four platforms: forums, Internet-Relay-Chat (IRC), DarkNet Markets (DNM), and carding shops (Benjamin et al. 2015). Among these, hacker forums allow users to freely share malicious tools designed to exploit vulnerabilities within popular systems and applications found in many modern organizations. Exploits found in forums have been used in numerous large-scale attacks. One notable example is the BlackPOS malware for the Target breach, which was found in the online hacker community months before the attack (Kitten 2014). Figure 2 illustrates a forum post with the BlackPOS malware attached for free download.



Figure 2. Forum Post with BlackPOS Malware

Despite their potential CTI value, hacker forums contain tens of thousands of unstructured, un-sanitized, multi-lingual text records. Standard CTI analytics (e.g., malware analysis) cannot handle these unique characteristics. Moreover, traditional analytics procedures (e.g., text mining, machine learning) require significant extensions to adapt to the CTI domain. The information-centric, application driven nature of the IS discipline makes IS scholars uniquely qualified to tackle these challenges. To date, IS researchers have leveraged their strengths to make contributions in four areas of cybersecurity: behavioral compliance, risk management, security investments, and market effects of cybersecurity (Hui et al. 2016). Studies within each category have used various paradigms across a multitude of cybersecurity contexts. Table 1 summarizes selected literature within each category.

Category	Year	Study	Focus	Paradigm
Behavioral Compliance	2015	Wang et al.	Insider threats within financial institutions	Behavioral
	2015	Vance et al.	UI design for user compliance	Behavioral
	2016	Chen and Zahedi	American and Chinese security perceptions	Behavioral
Risk Management	2010	Spears and Barki	User participation in security risk management	Behavioral
	2011	Chen et al.	Failures and diversification	Economic
Security Investments	2012	Li et al.	IT security controls	Economic
	2012	Ransbotham et al.	Markets for vulnerabilities	Economic
	2014	Kwon and Johnson	Security investments in healthcare	Economic
Market Effects of Cybersecurity	2012	Gupta and Zhdanov	Managed Security Services Networks	Economic
	2014	Kim and Kim	Malware resolution processes	Economic

Table 1. Summary of Selected IS Cybersecurity Literature (Adapted from Hui et al. 2016)

Behavioral compliance literature draws upon criminology and psychology theory to understand insider threats, UI design, and security perceptions. Risk management literature has aimed to strengthen organizational risk management by understanding user participation, failures, and diversification using behavioral and economic approaches. Security investments studies leverage economic approaches to substantiate the impact cybersecurity has within or across an industry (e.g., healthcare). The market effects of cybersecurity category focuses on how information security changes when it is placed in a market (e.g., outsourcing security to managed security providers).

Despite remarkable advances in each theme, many opportunities remain for IS scholars to contribute substantial cybersecurity knowledge, especially in CTI. IS scholars are well-positioned amongst other technical researchers (e.g., computer science) to make novel contributions in two critically needed aspects of CTI: understanding black hat hackers and analytics driven cybersecurity research (Chen et al. 2012; and Mahmood et al. 2010). Unfortunately, the current dearth of research output in both areas can have significant societal

ramifications, as hackers advance their capabilities at staggering rates. The behavioral and economic methods employed in current cybersecurity IS literature are not designed to execute the computationally intensive tasks of developing proactive CTI via hacker community analytics. Consequently, a set of guidelines for developing novel IT artifacts is needed.

Contrast with the behavioral and economic paradigms, design science focuses on the systematic development of novel IT artifacts to help solve practical problems of societal relevance (Hevner et al. 2004; Peffers et al. 2007). Hevner et al. (2004, p. 77) broadly defines IT artifacts as “constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems).” Irrespective of artifact, scholars argue that researchers should search and evaluate carefully select relevant problems from an application environment (i.e., domain), search for possible IT artifact design to address the problem, and rigorously evaluate their artifact. IT artifacts should contribute to the domain and IS knowledge base. IS knowledge base contributions can include situated artifact implementations, nascent design theory (e.g., novel design principles to inform future artifacts), and/or well-developed design theory (Hevner et al. 2004; Gregor and Hevner 2013). The breadth of IT artifact possibilities combined with the IS discipline’s diversity has led to four genres of design science research (Table 2): computational, optimization, representation, and economics (Rai 2017).

Genre	Description	Artifact Examples
Computational	Development of novel data representations, computational algorithms, business intelligence and analytics methods, and human-computer interaction (HCI) innovations	<ul style="list-style-type: none"> - Novel Support Vector Machine (SVM) kernels - Prototype systems (e.g., CyberGate)
Optimization	Solve decisional and operational issue with optimization methods	<ul style="list-style-type: none"> - Optimal rule combinations for product recommendations
Representation	Represent phenomena with methods, languages, and grammars	<ul style="list-style-type: none"> - Spatial and temporal constraints in conceptual database modeling
Economics	Designing mechanisms for the conduct of activities and activity exchange	<ul style="list-style-type: none"> - Competitive gaming model to inform the design of sustainable energy systems

Table 2. Genres of Design Science Research

CTI's emphasis on the rapid development of novel systems (e.g., honeypots), algorithms (e.g., malware analysis approaches), and visualizations (e.g., threat dashboards) indicates its reliance on consistent creation of novel IT artifacts of the computational design science genre. This genre offers three guidelines to inform IT artifact development (Rai 2017). First, artifacts can be informed by a relevant computational theory and/or pertinent domain knowledge when a strong theoretical foundation is lacking (e.g., deep learning). Second, researchers must demonstrate the technical superiority of their IT artifact by way of quantitative internal metrics (e.g., precision, recall, F1) and, when possible, external demonstration and/or validation (e.g., case studies). Finally, the IT artifact should contribute the appropriate knowledge (theoretical or methodological) to the IS knowledge base to support future research. These principles ensure IS researchers develop novel IT artifacts that add significant and relevant value to the application domain (e.g., CTI) and the IS knowledge base.

The limited body of computationally driven hacker community analytics to create proactive CTI both within the IS discipline and otherwise has resulted in organizations being unaware of what types of malicious exploits are available in hacker communities, what

vulnerabilities these exploits target, who key exploit sharing hackers are, and the emerging threats. Motivated by these gaps, my dissertation seeks to answer four broad research questions: (1) What exploits exist within hacker forums? (2) What vulnerabilities do hacker forum exploits target? (3) Who are the key hackers and exploit sharing communities in forums? (4) What are the emerging hacker exploits? Each essay in this dissertation answers one of the broad research questions listed above by developing a novel CTI computational IT artifact. Table 3 summarizes each essay in terms of its research question, method, algorithm, IT artifact, and artifact type.

Essay	Research Question	Method(s)	Algorithms/ Models	IT Artifact(s)	Artifact Type
I	What exploits exist within hacker forums?	Text/Data Mining; Social Network Analysis	Latent Dirichlet Allocation; Support Vector Machine	AZSecure Hacker Assets Portal	Prototype system
II	What vulnerabilities do hacker forum exploits target?	Text mining; Deep Learning	Deep Structured Semantic Model	Exploit Vulnerability DSSM; Device Vulnerability Severity Metric	Algorithm
III	Who are the key hackers and exploit sharing communities in forums?	Social Network Analysis; Deep Learning	Graph Convolutional Network; Autoencoder	Graph Convolutional Autoencoder	Algorithm
IV	What are the emerging hacker exploits?	Diachronic linguistics; deep learning; network analysis	Graph Convolutional Network; Autoencoder; Semantic Displacement	Diachronic Graph Convolutional Autoencoder framework	Algorithm

Table 3. Overview of Four Essays

The first essay sets the stage for the remaining three essays by asking the question “what exploits exist within hacker forums?” This essay presents a principled analytics framework employing state-of-the-art web, data, and text mining approaches to identify and

categorize hacker forum exploits. This framework powers a novel CTI system (i.e., this essay's IT artifact), the AZSecure Hacker Assets Portal. The Portal provides valuable search, sort, browse, and visualization functions such that CTI professionals within public and private sectors can develop in-depth intelligence of exploits available in online forums without directly accessing the platforms.

Despite essay I's important contributions, it has one fundamental issue: just identifying what hacker exploits exist is not enough for an organization. They must identify the most relevant exploits to them. This motivated essay II's core research question: "what vulnerabilities to hacker exploits target?" The IT artifact proposed in this essay, the Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM), leverages key text in vulnerability descriptions detected by state-of-the-art vulnerability assessment tools and exploit text to identify the most relevant hacker exploits for known vulnerabilities. This essay also presented the Device Vulnerability Severity Metric (DVSM). The DVSM incorporates the age of exploits and the severity of the vulnerabilities they link to enable device prioritization. EV-DSSM and DVSM's practical value was illustrated comprehensive case studies of eight major US hospitals and thousands of Supervisory Control and Data Acquisition (SCADA) systems worldwide.

Essay III looks beyond identification and linkages of exploits to answer a key question posed by law enforcement officials and many CTI professionals: "who are the key exploit hackers and exploit sharing communities in online hacker forums?" In this essay, the core operations of an emerging deep learning approach operating on graphs, the Graph Convolutional Network (GCN) were integrated into the popular deep learning architecture, the autoencoder. The resulting architecture, the Graph Convolutional Autoencoder (GCAE)

automatically generates embeddings of each hacker in an exploit sharing network based on (1) the exploit content they share and (2) their connections to other hackers. The GCAE's generated embeddings for each hacker are used to identify communities of exploit sharing members. A novel Exploit Degree Centrality (EDC) was designed to identify key hackers on a given exploit sharing network based on the age of their contributions. Both the GCAE and EDC's practical use was demonstrated with an in-depth case study of a large English hacker forum.

Essay IV is motivated by CTI professional's need for up-to-date knowledge by asking "what are the emerging hacker exploits?" To answer this, a novel deep learning framework, the Diachronic Graph Convolutional Autoencoder (D-GCAE) was developed. The D-GCAE operates on a text graph representation of hacker forum exploit text by incorporating custom graph convolution operations into the autoencoder architecture. Specifically, hacker exploit data was split into time-spells, text graphs were constructed in each, and the GCAE created word embeddings for words in each graph. Embedding alignment and semantic displacement measures adopted from diachronic linguistics literature map the shifts of exploit terms and identify emerging threats in forums. While D-GCAE's practical value was illustrated with a thorough case study of ransomware, the underlying algorithm can serve as an alternative approach to generating diachronic word embeddings to aid future IS research examining text generated from social media contexts.

2. ESSAY I: EXPLORING HACKER ASSETS IN ONLINE HACKER FORUMS

2.1. Introduction

Computer technology allows modern organizations to conduct their operations with a level of convenience and efficiency like never before. Unfortunately, many individuals with illicit cyber intent, also known as malicious hackers or cyber-criminals, often leverage dangerous cyber-tools or assets to conduct destructive cyberattacks against technologically driven organizations. Cyberattacks, or the deliberate exploitation of computer systems through the use of malicious tools and techniques such as Ransomware, Zeus Trojans, and Keyloggers, cost the global economy approximately \$445 billion per year and have negatively affected healthcare organizations like Premera Blue Cross, government entities such as the Office of Personnel Management (OPM), and large retail and consumer companies including Target, Home Depot, Sony, and Xbox Live (Elkind 2015; Kitten 2014; NSTC 2011; Riley et al. 2014; Sandle and Char 2014; Stuart 2014). The past few years have seen an unfortunate and disruptive growth in the number of cyberattacks (Granville 2015).

To help mitigate cyberattacks, companies such as FireEye and Cyveillance provide cyber threat intelligence (CTI) reports designed to help organizations protect against cyberattacks. To create their reports, these companies rely on data collected from actual attacks or events through mechanisms such as network logs, antivirus logs, honeypots, database access events, system login attempts, and IDS/IPS events (Shackleford 2015). The intelligence provided is reactive rather than proactive, as it is based on data from attacks that have already happened. The reports do not provide intelligence on tools that hackers have developed but not yet used for cyberattacks that also have the potential to cause great damage (e.g., zero-day attacks). Additionally, these reports often ignore the specific actors who are responsible for

such exploitations, resulting in an incomplete picture of the overall hacker ecosystem (e.g., communities, motivations, specialties, tools, etc.). These shortcomings have led industry leaders such as Ernst and Young to note that traditional CTI is not “sufficient to properly address risk in individual organizations” and that “organizations need to take a more proactive approach to cybersecurity” (EY 2014).

To address some of the faults in traditional CTI, industry and academia alike have emphasized the need to develop more comprehensive and proactive CTI by directly collecting, identifying, and analyzing data from the online hacker community to better understand malicious tools and individuals (Mahmood et al. 2010; McMillan 2012). While the online hacker community contains a variety of components (e.g., underground economies, IRC channels, etc.), hacker forums in particular provide the technical mechanisms for hackers to easily provide and acquire freely accessible, malicious tools such as Zeus, Ransomware, SQL injections, and DDoS, amongst others (Benjamin and Chen 2012; Benjamin et al. 2015; Samtani et al. 2015). Overall, there are hundreds of hacker web forums across geo-political regions such as the US, Russia, and China. The forums contain tens of millions of postings and cover a wide variety of topics, such as underground economies, data breaches, and cyberwarfare. Also included are tens of thousands of malicious assets created by millions of hackers, many of which pose great danger to organizations. Given the hacker forum scale and that hacker forum tools have been used to attack organizations (e.g., BlackPOS code was available in forums months before the Target attack), hacker forums can also serve as viable data sources for understanding emerging tools and assets (Benjamin et al. 2016; Kitten 2014; Motoyoma et al. 2011; Samtani et al. 2015) and can contribute to more accurate, proactive, and comprehensive CTI.

Directly collecting and analyzing large amounts of hacker forum data presents unique technical challenges that limit researchers' abilities to produce comprehensive CTI studies. These challenges include vast amounts of textual data, robust anti-crawling measures, foreign language barriers, little-known hacking terms, and complex forum structures. However, given the recent interest in gathering intelligence directly from hackers and the growing emphasis of security Big Data studies in information systems literature (Abbasi et al. 2015; Chen et al. 2012), one purpose of this research is to contribute to information systems literature by developing a novel CTI framework leveraging principled web, text, and data mining techniques to methodically collect, identify, and analyze malicious hacker assets in underground forums. Specifically, we analyze these assets by applying state-of-the-art techniques such as Support Vector Machine (SVM) and Latent Dirichlet Allocation (LDA) to identify and understand the implementations of such assets, and leverage rich forum meta-data to understand the key trends of malicious cyber-assets. We also employ bipartite social network analysis techniques to identify key hackers for selected malicious assets. The practical results of this research have the potential to mitigate future cyberattacks, thus reducing the overall cost of cybercrime on the global economy. Additionally, the tools that are identified in this analysis can be presented through a cybersecurity education portal designed to help cybersecurity educators and students enhance their understanding of malicious tools and assets.

2.2. Literature Review

2.2.1. Hacker Community Research

Hackers congregate on a variety of online platforms such as IRC channels, carding shops, and hacker forums to exchange content and knowledge (Benjamin et al. 2016; Benjamin et al. 2015). While all of these platforms are rich data sources for research, hacker forums are

a unique and particularly fruitful platform for identifying malicious assets as they allow users to easily and systematically post, save, and retrieve certain types of content (e.g., source code, malicious files) which other online platforms do not (Benjamin and Chen 2012; Benjamin et al. 2016; Benjamin et al. 2015). Additionally, forums have seen a consistent level of usage by hackers over a long period of time and have broad geo-political coverage and topical interests (Benjamin et al. 2015). As a result, researchers have the potential to conduct large-scale, diverse, longitudinal analyses. In general, US forums focus primarily on cybercrime and general hacking, Russian forums on underground economies and data breaches, and Chinese forums on cyberwarfare and virtual goods (Goel 2011). Various subforums within the overall forum are dedicated to specific subareas. For example, a hacker forum specializing in malware may have subforums dedicated to Ransomware or Zeus code. Within each subforum, members can create and post in specific discussions (i.e., threads). In general, the subforum and thread titles contain specific keywords or phrases which allow users to identify the purpose of each area such that they can post and acquire content. While the specific posting privileges and mechanisms vary based on forum and forum standing, users can generally share hyperlinks, pictures, videos, source code, attachments, and other resources with each other in these threads. Figure 3 illustrates how users can navigate from the overall forum to a thread where they can acquire and post malicious source code within the popular hacker forum OpenSC.

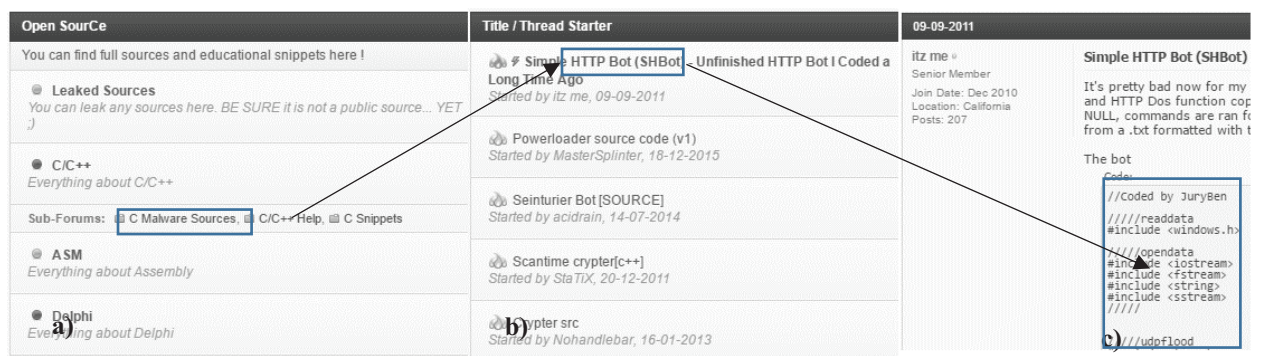


Figure 3. How to Access Exploits

(a) From the main forum page (far left) a user can access the “C Malware Sources” subforum. (b) The malware sources link takes the user to a page listing numerous threads providing access to malicious source code. (c) The link labeled “Simple HTTP Bot” takes the user to a page with an HTTP Bot designed to flood a network.

Researchers who are interested in studying such forums still face numerous technical challenges. Many hacker forums have rigorous vetting processes for prospective members, have postings often in foreign languages, contain unfamiliar hacking related terms, and inconsistent subforum structures that make categorization of content difficult. In addition, many forums employ sophisticated anti-crawling measures which make comprehensive data collection difficult. These technical challenges have generally resulted in researchers manually collecting small sets of data and applying qualitative techniques to understand the social interactions and networks of hacker community members. Though these techniques suffer from limits on their scalability, such studies provide valuable insight into the behaviors of hacker forum members, noting that regardless of geo-political orientation (Russian, English, etc.), the majority of forum participants are unskilled and only a small percentage of hacker members possess a high level of skill in creating and disseminating malicious assets (Chu et al. 2010; Holt et al. 2012; Holt 2013; Motoyama et al. 2011; Radianti 2010; Yip 2011). Additional studies using automated techniques such as deep learning-based sentiment analysis, Interaction Coherence Analysis, and regression analysis have discovered that the quality and volume of hacker assets a member posts are key factors behind their reputation and forum standing (Abbasi et al. 2014; Benjamin and Chen 2012; Li et al. 2016).

While it is clear that hacker assets help drive hacker reputation and standing, only a few studies have tried to identify what these tools or assets actually are. These studies have primarily used manual explorations or interviews with a Subject Matter Expert (SME) to identify hacker forum assets. Hacker forum assets usually come in three main forms: source

code, attachments, and tutorials (Samtani et al. 2015; Samtani et al. 2016; Samtani et al. 2017). Source code is code written in a programming language embedded within a forum posting. Hacker forum code, much like code found in the popular forum Stack Overflow, is generally raw, incomplete, and cannot be executed without a programming environment. Hacker source code can range from benign, general purpose programming code (e.g., web development code) to more malicious, damaging code such as SQL Injections and Zeus Trojans. Attachments, on the other hand, are files attached to forum postings. These may be harmless files (such as general programming books) or malicious executables and tools (such as Ransomware). Finally, tutorials are forum posts designed to educate other hackers on specific topics (for example, creating a cyber-weapon or conducting a phishing attack). Generally, the post content found with the asset is descriptive of the purpose of the asset.

Ablon et al. (2014) discovered that payloads, full services, and credit card information is available in hacker black markets, while Chu et al. (2014) and Samtani et al. (2015) found freely accessible SQL injections, banking vulnerabilities, and Zeus Trojans in hacker forums. Ablon et al. (2014) further noted that malicious code assets are the most abundant asset within hacker forums and are often used to facilitate cybercriminal activities. However, given that most of these studies leveraged manual techniques or interviews with SME's, they cannot be scaled to larger amounts of data. Furthermore, they do not leverage the rich forum metadata (e.g., author names, post dates, etc.) available with the forums to conduct deeper analyses or systematic identification of all malicious hacker assets. Given these limitations and the inherent strengths of hacker forum platforms, we next review cyber threat intelligence literature to gain perspective on the data sources and the analytical procedures used to create valuable intelligence.

2.2.2. Cyber Threat Intelligence (CTI)

The SANS Institute defines cyber threat intelligence as “threat intelligence related to computers, networks, and information technology” (Farnham 2013). CTI has traditionally been motivated by the desire for companies and individuals to better protect their cyber-infrastructure from an attack (Farnham 2013; Shackleford 2015). Today, many vendors, such as Symantec, McAfee, Trend Micro, FireEye, Cyveillance, Sophos, and Kaspersky, have established themselves as leaders in developing and providing CTI to other entities in the government and commercial spaces. Many companies also support their own internal CTI divisions to produce customized reports. Regardless of company, data is generally gathered from a variety of sources including network traffic logs, database access events, configuration modification logs, IDS/IPS events, login and access logs, external activity to commonly hacked ports (e.g., 1080, 21, 22, 23, 3306, etc.), honeypot logs, antivirus logs, etc. (Shackleford 2015). While these data sources are relatively comprehensive in terms of the network data gathered, their inherent weakness is that they do not integrate information from or about the actual communities from which the attacks originate (a data source which can contain a vast amount of valuable data), and instead rely on their own systems to log actual cyberattacks (Shackleford 2015). In addition, analyses conducted on the collected data generally opt for the calculation and visualization of basic descriptive statistics, often in the form of real-time streams, dashboards, and reports. For example, many CTI reports contain information about the malware variants which were picked most frequently by their sensors, various statistics about network traffic, suspicious URLs to monitor, and blocked attacks (Shackleford 2015). Some reports also leverage attacker IP addresses gathered by honeypots and other network logging software to provide geographical overviews of countries conducting specific types of attacks.

Temporal analyses may also be used to illustrate the growth or decline in specific types of malware or certain cyberattacks.

Though there is great value in the prevailing CTI process and reporting style, existing CTI reports have been criticized for being too high-level and for being merely reactive to already known threats (EY 2014). Furthermore, the reports do not detail the specifics of the exploits, such as implementation methods or programming language. Such information has proven to be valuable for companies: knowing the method of implementation for an exploit can aid in building more robust cyber-defenses. Unfortunately, given that traditional malware analysis techniques generally look at one malware binary at a time, the sheer volume of hacker assets, and the fact that the majority of assets in hacker forums are source code (Samtani et al. 2015), traditional malware analysis methods cannot be used. However, the techniques detailed in source code analysis literature can be adopted for useful and advanced hacker source code analysis.

2.2.3. Source Code Analysis

Source code provides content and structure advantageous to the discovery of its technical implementation (i.e., programming language) as well as its purpose. Given the large amount of source code in hacker forums, we review and borrow techniques from two sub-areas of source code analysis literature: source code classification and source code topic extraction. Both applications have been used extensively in the context of mining software forum repositories.

Source code classification research is motivated by the desire to improve software reuse and organization and focuses primarily on classifying code in online repositories (e.g., SourceForge or Ibiblio) into predefined categories such as databases, games, email, attack

vectors, or other domain specific areas (McMillan et al. 2011; Ugeral et al. 2002). The general approach to classifying source code is to identify target classes for classification, develop a training set with sample source code from each class, and use textual features to train the classifier for unseen code. Various classifiers have been evaluated, with SVM consistently showing the highest performance (McMillan et al. 2011; Ugeral et al. 2002). Code can also be classified into programming languages by training a classifier on sample source code from several different languages and applying the trained classifier to unseen code files to classify them into their designated languages. Prior work has found that SVM classifiers using term frequencies as features have the most effective performance for this task (Ugeral et al. 2002). Even so, these classifiers do not identify the function the code serves unless it is already in a pre-defined domain or manually executed. However, a second sub-area of source code literature (code topic extraction) focuses on using topic modeling techniques such as Latent Dirichlet Allocation (LDA) to automatically identify topics and applications of large amounts of source code whose purposes are unknown (Chen et al. 2015).

LDA is a robust generative probabilistic model used to automatically discover latent topics within large text corpora (Blei et al. 2003). Based on a set of parameters (pre-defined number of topics to be extracted, iterations, etc.), LDA extracts a set of topics from a collection of textual documents. Each topic is a distribution of words, and each document is a mixture of corpus-wide topics. Generally, modeling a large number of topics produces fine-grained results, while modeling fewer topics returns coarser outputs (Baldi et al. 2008; Barua et al. 2014).

Compared to classification models, LDA lacks rigorous quantitative internal and external validation metrics partially due to its unsupervised nature. LDA is often evaluated

through two standard techniques: through manual evaluation, and through a statistical metric, perplexity (Blei et al. 2003). In the manual approach, the outputted topics are qualitatively evaluated for their coherence and clarity by several experts, with inter-rater reliability being a commonly used technique to ensure concordance between topic evaluators (Chen et al. 2015). On the other hand, perplexity is a quantitative measure designed to mathematically calculate the optimal number of topics to model for. To calculate perplexity, a small set of the text corpus is held out as testing data, while the majority of the corpus is used to train the LDA model. Perplexity measures the likelihood that the held-out data is truly generated from the underlying topic distributions based on the inverse log likelihood of the held-out documents. Lower perplexity score indicates better topic model match (Blei et al. 2003).

Although LDA is traditionally applied to natural language documents, it has also been used to discern the topics in and purpose of code in code repositories such as Stack Overflow (Allamanis and Sutton 2013; Bajaj et al. 2014; Barua et al. 2014), SourceForge (Baldi et al. 2008; Linares-Vásquez; Linstead et al. 2008; Wang et al. 2013), or large software systems such as Hadoop or Petstore (Grant et al. 2011; Maskeri et al. 2008; Tian et al. 2009). However, source code is not natural language and any comments or post content must be pre-processed before adopting LDA. Standard pre-processing steps include special character removal (e.g., quotes, hyphens, underscores, etc.), identifier splitting (e.g., “dataAuthResponse” becomes “data Auth Response”), case-folding (e.g., “Response” to “response”), and stopword removal (Chen et al. 2015). After pre-processing, studies typically extract between 40 and 150 topics from the code. Topics are manually labeled after extraction. However, the majority of these studies did not use perplexity to statistically evaluate the optimal topic number to model for, instead opting for manual evaluation (Allamanis and Sutton 2013; Bajaj et al. 2014; Barua et

al. 2014; Linares-Vásquez et al. 2014) or no evaluation (Baldi et al. 2008; Linstead et al. 2008; Maskeri et al. 2008; Savage et al. 2010; Tian et al. 2009).

Although LDA can extract topics from code, it cannot identify who disseminates the code in a forum context. However, given that postings have explicit author metadata, the posts related with certain topics from the LDA results can be paired with Social Network Analysis (SNA) visualization and statistical measures to understand key individuals for particular subjects. Indeed, this combination of techniques has been used to great success in other literature studying the Dark Web and other forum contexts (L'huillier et al. 2010; Rios et al. 2011).

2.2.4. Social Network Analysis (SNA)

SNA builds on graph theory to study the relationships between social entities and the implications of these relationships. Social networks are represented with two major components: nodes and edges. Nodes represent social entities such as individuals, organizations, social media members, etc. Edges represent the communication ties, relations, or linkages between nodes, and can be directed (one-way relationship) or undirected (mutual, two-way relationship).

Traditional SNA is useful for studying phenomena such as technology diffusion between individuals, but is less applicable in determining the relationships between two different types of entities, such as hackers and assets. However, an affiliation network (also known as a two-mode or bipartite network) is a special type of network that can model such relationships (Faust 1997). This type of network partitions nodes into two sets: individuals and events, wherein one node type (individuals) is “affiliated” with the other node type (events). Relationships in this setup are directed and are modeled between the nodes, but not within.

This means that nodes representing individuals cannot have a relationship with other individuals, only with events. Bipartite networks are useful when trying to observe relationships between different types of nodes (i.e., between individuals and events). Bipartite networks are used in a variety of contexts such as Wikipedia co-authorship (Keegan et al. 2012), knowledge translation in health forums (Stewart and Abidi 2012), and sexual relationship networks (Fujimoto et al. 2013). Despite their flexibility in modeling different contexts, the main disadvantage of bipartite networks is that they can generate only a subset of the well-known network metrics (Faust 1997). To calculate all network metrics (e.g., distance, length, diameter, and radius) and node-level metrics (e.g., degree, closeness, betweenness, eigenvector), bipartite networks are often projected as monopartite networks (i.e., networks with only one node type).

2.3. Research Gaps and Questions

Several research gaps were identified from our literature review of hacker communities, cyber threat intelligence, source code analysis, and social network analysis research. Hacker community research has focused on general hacker interactions and the identification of key hackers within communities. Few studies have examined hacker forum content or assets. Moreover, these studies' approaches have primarily been manual and qualitative, and not scalable or comprehensive enough for analyzing a large volume of assets. Previous studies have also not leveraged forum metadata to conduct deep, comprehensive analyses for valuable CTI. CTI reports are based primarily on data from attacks that have already occurred, and do not consider attackers' ecosystems. As a result, the reports are reactive to current threats in cyberspace rather than proactive in identifying future threats. Source code literature heavily emphasizes classification and topic modeling of large amounts

of code found in online software repositories or systems, but few studies have analyzed hacker community source code. Finally, social network analysis has been used sparingly to depict hacker relationships and interactions. Furthermore, although traditional approaches are useful for identifying relationships between hackers, they do not fully represent the relationships between hackers and assets. While variations of the traditional analysis (i.e., bipartite networks) can help in this regard, we were unable to find research leveraging these techniques and their associated centrality measures to mathematically or visually represent the relationships between hackers and assets. Based on these research gaps and the desired research outcomes, the following research questions are proposed for this study:

- What types of hacker assets are available in underground communities?
- What are the characteristics and functions of these hacker assets?
- Who are the key hackers disseminating hacker assets in underground forums?

This essay addresses the research gaps by creating a principled, automated, and scalable framework for proactive CTI. This framework collects a large and novel dataset of hacker assets directly from the hacker community and utilizes state-of-the-art topic modeling and classification techniques with forum metadata to systematically identify and gain a deep understanding of this large volume of assets. Bipartite social network analysis techniques are also used to computationally identify key hackers for selected assets. Such analysis addresses some of the current limitations present in traditional malware analysis and CTI methodologies.

2.4. Research Design and Testbed

Our hacker asset analysis framework (Figure 4) comprises three major components: a data collection and data pre-processing component, an asset analysis and evaluation section,

and a social network construction component. These components are detailed in the following subsections.

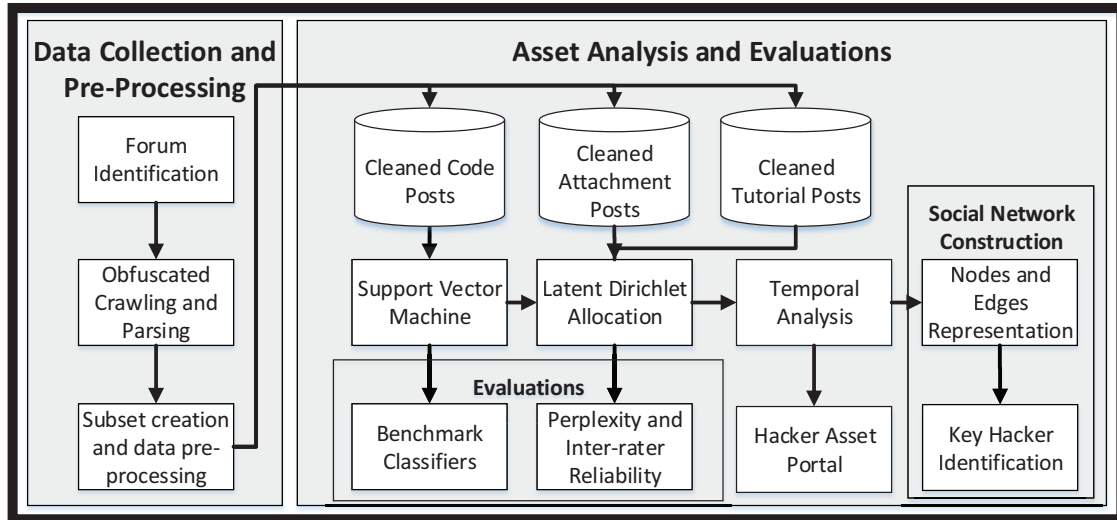


Figure 4. Methodological Research Framework for Analyzing Hacker Assets

2.4.1. Data Collection and Pre-Processing

Similar to prior hacker forum literature, one English and six Russian hacker forums were identified for collection and analysis. These seven forums were selected from among hundreds of hacker forums for several reasons. First, these forums were suggested for examination by several cybersecurity experts. Second, English and Russian hacker communities are notorious for creating and using malicious cyber-assets for cyberattacks (Holt 2013). Third, these forums can be accessed without an invitation, thus reducing the risk of researcher identification. Finally, these forums are well-known in the hacker community for containing a plethora of malicious assets.

To circumvent the forums' anti-crawling mechanisms such as checking of user-agents, username and password authentication, timing out of sessions, etc., a Tor-routed web crawler using forum credentials downloads and stores all forum HTML web pages onto local hard disks for offline browsing and further processing. Since forums create special HTML tags and

structures every time a user posts an asset (source code, attachment, or tutorial), we utilize a specialized parser to identify such structures and parse all of the post, asset, thread, and author data into the database. After collection and parsing, the source code, attachment, and tutorial posts are placed into separate subsets using SQL queries. If a post contains any source code, it is placed in the code subset. Regardless of subset, we utilize Google Translate to translate all posts to English, remove special characters, split identifiers (e.g., readFile to read File), fold case, and remove stop-words. Table 4 summarizes the final collection. To ensure we protect ourselves from potential attacks from hackers within these communities, all forum names are anonymized.

Forum	Language	Date Range	# of posts	# of threads	# of authors	# of source code snippets	# of code posts	# of tutorials	# of attachments
O****C	English	2/6/2005-9/15/2015	124,993	16,046	6,796	13,155	5,478	1,480	2,776
D****b	Russian	11/4/2004-9/15/2015	5,903	648	1,007	506	138	14	27
E****N	Russian	2/26/2005-9/15/2015	157,106	16,194	7,761	6,341	3,037	336	N/A
P****c	Russian	8/26/2006-9/15/2015	25,865	5,092	2,276	734	389	161	N/A
R****u	Russian	8/3/2009-9/15/2015	4,998	758	243	4,063	370	72	N/A
X****k	Russian	4/15/2009-9/15/2015	50,337	14,026	3,827	2,009	1,536	202	172
X****c	Russian	6/6/2007-9/15/2015	62,316	48,947	18,462	1,681	719	147	5
TOTAL:	English/ Russian	11/4/2004– 9/15/2015	431,518	101,711	40,372	28,489	11,667	2,412	2,980

Table 4. Summary of Research Testbed

The collection contains 431,518 posts in 101,711 threads made by 40,372 authors between November 4, 2004 and September 15, 2015. The Russian forum ExploitIN and the English forum OpenSC have the most posts with 157,106 and 124,993, respectively. From the entire collection, the parsers and queries identified 11,667 source code posts with 28,489 code snippets (a source code post can have multiple code snippets), 2,412 tutorials, and 2,980 attachments. However, it should be noted that the majority of these attachments (2,776 of the total 2,980 attachments) are from OpenSC. Three forums, ExploitIN, Prologic, and Reverse4You, do not allow members to post attachments, while DamageLab, Xakepok, and Xeksec limit the privilege of posting attachments to more senior forum members. Nevertheless, these forums were still analyzed as they contained relatively large amounts of other assets: source code and tutorials.

2.4.2. Asset Analysis and Evaluations

The asset analysis component aims to automatically understand the nature and implementation of the pre-processed source code, attachment, and tutorial posts. First, the source code is classified by coding language to provide insight into their implementation (e.g., what language is used to program formgrabbers). This classification also facilitates better overall code organization for educational purposes. Ten classes were selected for classification based on observations of the most used code in the forums and in past research: Java, Python, C/C++, HTML, PHP, Delphi, Visual Basic, SQL, Ruby, and Perl (Samtani et al. 2015; Ugeral et al. 2002). One hundred code files for each language were used to train an SVM classifier (using term frequencies as features) with RapidMiner's LIBSVM package. These files were selected based on their uniqueness and exclusivity to each language (e.g., JSoup only appears in Java), as that helps to increase the effectiveness of the classifier (Ugeral et al. 2002). After

training, the SVM classifier was evaluated against other benchmark classifiers using metrics such as accuracy, precision, recall, and F-Measure. Paired t-tests are also conducted. Once evaluated, the trained classifier is applied to each forums' source code.

After classification, LDA is run on each forums' code. As with previous literature, all comments and post content are left in to provide extra context during LDA analysis (Chen et al. 2015). Including the raw source code along with the comments and post content has shown to significantly increase LDA's performance as compared to just the post content or comments (Chen et al. 2015). Attachment and tutorial posts, however, do not need any sort of classification and are descriptive of the file or instructions provided in the post subject field (e.g., BlackPOS attachment and malicious document tutorial from earlier). As such, LDA is run directly on these subsets after pre-processing. For the purposes of this study, we include all assets to gain a comprehensive perspective of the entire forum. This understanding can lead to additional research inquiries that are not apparent by limiting our analysis to a subset of the data.

Regardless of asset, perplexity is calculated for each subset separately at varying epochs (e.g., 5, 10, 15, 20, etc.) to identify the appropriate number of topics to extract for each subset. The model providing the lowest perplexity, thus indicating the highest level of performance, is used. We manually label the topics based on our interpretation of the top 15 keywords which are outputted from the trained model. After extracting the appropriate number of topics from the LDA model based on the perplexity calculations, we asked a panel of six cybersecurity students to validate our interpretations of the outputted topics. In this task, we provided each of the students with the topic keywords for each topic, our interpretation of each topic (i.e., topic label), and some sample postings which were categorized falling into the topic.

Each panelist was asked to agree or disagree with our interpretation of the topic based on the provided keywords and postings. If the panelist disagreed with our interpretation of the topics, we asked the panelist to provide an alternate suggestion for the topic label based on the keywords and the provided postings. To limit biases, we asked each of the raters to rate the topics independent of other raters. Consistent with standard practice, we calculate the level of concordance between the raters using the Cronbach's alpha statistic. Selected topics for malicious assets are then temporally visualized.

After all analyses, we developed a web portal, the Hacker Asset Portal (HAP), to host selected assets. As mentioned in the introduction, providing these assets for the larger cybersecurity community can provide a novel approach to enhancing current cybersecurity education. Students and cybersecurity professionals alike would gain the ability to learn how the hacker community is developing their malicious tools. Such knowledge would aid in developing and implementing more robust cyber-defenses to block potential attacks. Given these benefits, the HAP aims to provide selected users within the cybersecurity community the ability to browse, search, sort, and download assets. We also create a visualization system allowing users to identify asset trends and key hackers for assets (based on post frequency).

2.4.3. Social Network Construction

After identifying malicious topics, the metadata associated with selected topics' posts are used to build social networks to identify key hackers for those topics. For a specific topic, we extracted all of the posts which had the highest probability of belonging to that topic, as calculated by LDA. Using the associated thread and author data for each post, we construct bipartite networks connecting hackers (node type 1) to threads with specific types of assets (node type 2). Doing so is consistent with prior literature (L'huillier et al. 2010; Rios et al.

2011). While directly modeling hacker relationships with asset posts is preferable, such information is limited to forum administrators. Furthermore, this type of bipartite network configuration has been used in various other forum contexts (Stewart et al. 2012). Given the limited statistical analysis which can be conducted on bipartite graphs, we further project the graph into two monopartite graphs; one representing malicious hacker tools, the other representing hackers. From here, we identify topological properties of each network by running global network statistics such as network diameter, density, connected components, and average path lengths. We gain a more granular understanding of key hackers by calculating degree and betweenness centrality measures.

2.5. Results and Discussion

This section first presents SVM classification and LDA evaluation results. Subsequently, interesting results for source code, attachment, and tutorial assets are highlighted. Finally, the social network analysis results identifying key hackers for specific topics is presented.

2.5.1. Classification Performance

In this experiment, we aim to establish a baseline of classification performance based on prevailing classification techniques. We evaluated several state-of-the-art classifiers against SVM: k-Nearest Neighbor (k-NN), Naïve Bayes, and Decision Tree using ten-fold cross-validation. In this approach, the training data (i.e., the 1,000 source code files from online repositories) is partitioned into 10 disjoint subsets. The classifier is trained on nine of these subsets and tested on the remaining subset. This process is repeated until each of the 10 subsets partake in both training and testing. The accuracy, precision, recall, and F-measure scores are summarized in Table 5.

Classification Algorithm	Overall Accuracy	Precision	Recall	F-Measure
Support Vector Machine	98.20	96.36	98.20	98.28
k-Nearest Neighbor	64.00	83.47	64.00	72.24
Naïve Bayes	86.00	88.57	86.00	87.26
Decision Tree	82.60	86.41	82.60	84.42

Table 5. Classification Results

Overall, SVM, Naïve Bayes, and Decision Tree demonstrated high accuracy, precision, recall and F-measure. Further statistical tests show that SVM significantly outperformed the other classifiers (results of paired t-tests are reported in Table 6), even with their strong performances in other metrics. These findings are consistent with other code classification literature (Linares-Vásquez et al. 2014; McMillan et al. 2012; Ugerel et al. 2002). Given SVM’s strong statistical performance, it is adopted as the primary classifier for our subsequent analyses.

Metric	SVM vs k-Nearest Neighbor	SVM vs Naïve Bayes	SVM vs Decision Tree
Precision	<.0001***	<.0001***	.000102**
Recall	<.0001***	<.0001***	<.0001***
F-Measure	<.0001***	<.0001***	<.0001***

Table 6. p-Values for Pair-wise t-tests for SVM against benchmark classifiers

2.5.2. LDA Evaluations

In addition to evaluating the efficacy of the SVM classifier, we also calculate the perplexity at various epochs (5 to 100, incrementing by five) to identify the optimal number of topics for which to model each subset of data (Table 7). After extracting the optimal number of topics, our raters independently validated our interpretation of the topics based on the provided topic keywords, our topic labels, and postings within the given topic. Inter-rater reliability calculations demonstrated that our raters reached a 0.9393 Cronbach’s alpha score, indicating a high level of concordance between all raters.

Data	Optimal Topic Number	Perplexity
Attachments	100	1794.184
Tutorials	90	2,150.418
D****b	60	440.772
E****N	65	1,424.834
O****C	95	4,866.838
P****c	95	970.041
R****u	80	1,576.980
X****k	90	390.453
X****c	80	1,198.133

Table 7. Optimal Topic Numbers

2.5.3. Malicious and Emerging Source Code Topics

Source code assets are particularly valuable to hackers as they can be easily adjusted or extended to incorporate new functionalities, given the hacker’s technical ability to do so. Our code analysis reveals each forums’ malicious code, their implementations (i.e., language), and temporal trends. For illustration purposes, we present results of the three of the largest forums in our collection (in terms code assets), OpenSC, ExploitIN, and Reverse4you as case examples. OpenSC and ExploitIN are also two of the longest-running forums in our collection, with both forums starting in 2005.

Source Code Topic Label	Primary Language of Implementation	% of topics in OpenSC	Keywords in topic (s)
Crypters	Java	2.10%	Decrypt, encrypt, encrypted, key, generate, algorithm, keys, polymorphism
Metasploit Exploits	Ruby	1.05%	Metasploit, framework, applications, install, exploit, systems, advanced, analysis, compile
Shellcode Exploits	C/C++	1.05%	Shellcode, x30, x20, x65, x0a, x6f, x6e, x40 (*this is sample shellcode that appeared in topic*)
Backdooring Websites	PHP	1.05%	Backdoored, http, com, html, source, org, php, index, htm, dump, log, script
Memory Injections	Delphi	1.05%	GetProcAddress, process, hprocess, mem, kernel32, pmem, inject, pointer, injectstring
SQL Injections	SQL	1.05%	Php, http, post, url, get, mysql, login, upload, password, select, sql

Process Injections	Delphi	1.05%	Process, pid, thread, injection, processed, detach, target, library, list, dllmain
Keylogging	C/C++	1.05%	Keylogger, password, http, rar, firefox, upload, users, stealer, subject
Total:	-	9.45%	-

Table 8. Malicious Source Code Topics in OpenSC Forum

9.45% of the topics in OpenSC are considered malicious, while the remaining 90.55% were benign. While still interpretable LDA results, these benign topics were often general and did not pertain to exploitations or malware. For example, some of the benign topics include general Java or Python programming. Despite the large amount of such assets, OpenSC members can still freely access a great variety of malicious code, including, for example, low-level system exploits such as shellcode, memory, and process injections, all designed to harm a computers' memory and system processes; application exploits such as crypters, designed to encrypt user files, and keyloggers, built to steal sensitive user data; and web attacks such as backdooring websites and SQL injections, intended to exploit webpage vulnerabilities. The malicious source code identified by our classifier is listed in Table 8. Our classifier further revealed that these exploits are written in languages ideal for the system or target for which they are designed. For example, memory injections, shellcode exploits, and process injections were primarily written in Delphi or C/C++, Metasploit modules in Ruby, SQL injections in SQL, and for website backdooring, PHP. In addition to classifying malicious source code topics, the forum timestamp meta-data was leveraged to create a line chart plotting the proportion of posts related to a particular topic for each year in Figure 5.

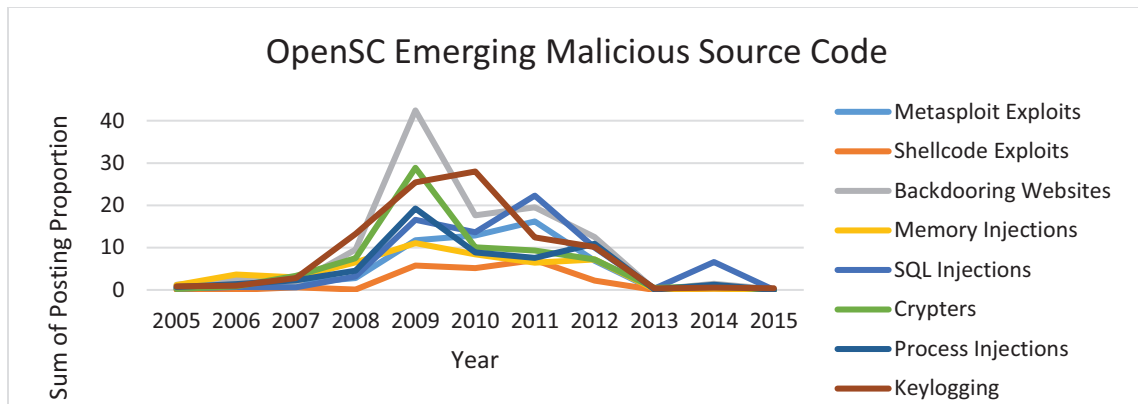


Figure 5. Emerging Malicious Source Code Topics for OpenSC

Figure 5 shows that many of the malicious source code topics were popular between 2009 and 2011. Some of the peaks shown on the chart, such as backdooring websites and keylogging, are consistent with well-known events. In 2009, for example, the popular website builder Wordpress received widespread attention for being vulnerable to backdoors (Motoyama et al. 2011). Additionally, there was a surge in real keylogging exploitations between 2008 and 2010 (HelpNetSecurity 2009). More recently in the forum, crypters have received slightly more attention, possibly because they are the foundation for the robust cyber-weapon Ransomware (a tool that encrypts a users' files as ransom for payment), one of the most malicious tools known to date (Symantec 2014). Such an increase in forum assets correlates and is validated with the growing amount of actual exploits conducted with crypters and Ransomware during the same time period (Symantec 2014). The chart also shows a decrease in popularity for all malicious codes over time. The likely explanation for this decrease is that OpenSC has seen a decline in uptime in recent years. If the forum was not online, its users would have been unable to post forum content.

ExploitIN also contains a number of network, web, and application exploits for its members to freely access. Interestingly, SQL injections and shellcode exploits in ExploitIN were both implemented in the same languages as the SQL injections and shellcode exploits

found in OpenSC. In addition to these exploits, ExploitIN also contains network binders, spam services, password cracking tools, and banking rootkits for their members to access. Table 9 summarizes these topics.

Source Code Topic Label	Primary Language of Implementation	% of topics in ExploitIN	Keywords in topic (s)
Shellcode Exploits	C/C++	6.00%	Buffer, int, shellcode, overflow, argv, stack, xff, x40, exploit, vulnerable
SQL Injections	SQL	3.00%	Select, sql, table, error, database, userse, injection, query, null, union, request
Network Binders	C/C++	1.50%	Bind, router, network, utility, information, interface, scanner, command, tool, cisco
Password Cracking	Python	1.50%	Password, login, proxy, user, pass, username, get, type, log, auth, set
Spam Services	Java	1.50%	Spam, optimization, site, traffic, engines, website, url, page, registration, visitors
Banking Rootkits	Java	1.50%	Rootkit, bank, malicious, security, network, malware, banks, infected, tools
Crypters	Java	1.50%	Crypt, decrypt, encrypt, encrypted, file, install, kriptor, keys
Total:	-	16.50%	-

Table 9. Malicious Source Code Topics in ExploitIN Forum

In analyzing the temporal trends for ExploitIN assets in Figure 6, we discover several interesting findings. First, we identified more recent, emerging trends as compared to OpenSC, specifically for network binders, password crackers, spam services, crypters, and shellcode exploits. Additionally, we discovered that some of the trends present in OpenSC (e.g., SQL injection popularity in 2009-2011) were also in ExploitIN. This shows that the growth in popularity for a specific asset during a given time period may not be limited to a single forum, but may be a cross-forum phenomena. This is further illustrated with crypters showing a recent increase in popularity (2014-2015), coinciding with both the growth in OpenSC as well as recent Ransomware events.

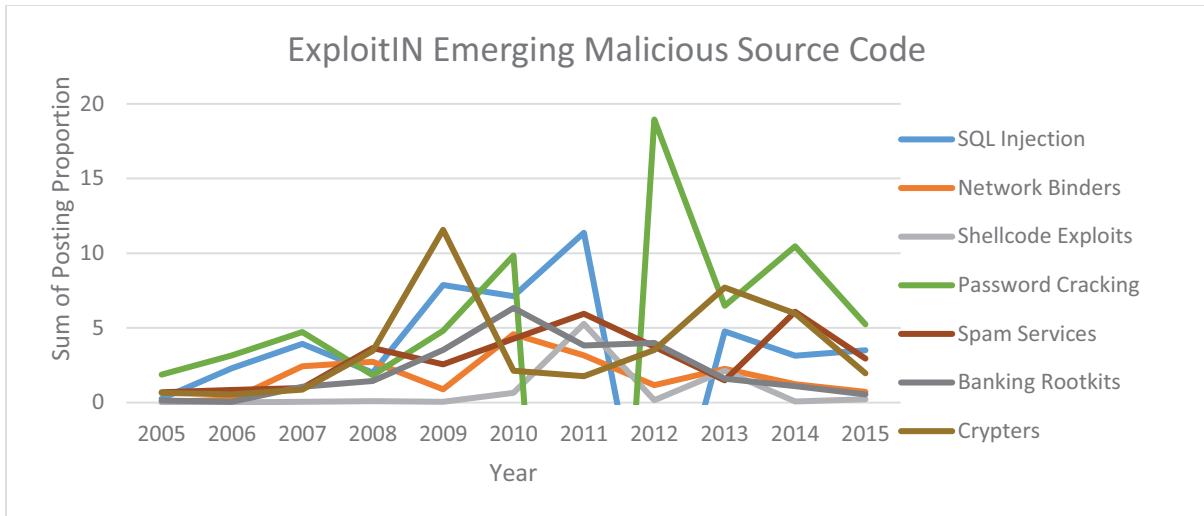


Figure 6. Emerging Malicious Source Code Topics in ExploitIN

Another forum, Reverse4You, contains assets not seen in either OpenSC or ExploitIN. For example, many bruteforcers, or tools created to deliberately attempt to break through login systems by enumerating through sets of usernames and passwords were available for access. In addition, there were also a variety of Dynamic Link Library (DLL) Exploits available. Such exploits are targeted at directly at critical code libraries for the Windows operating system. Table 10 summarizes the malicious assets available in Reverse4You.

Source Code Topic Label	Primary Language of Implementation	% of topics in Reverse4You	Keywords in topic (s)
Shellcode Exploits	C/C++	10.0%	Stack, pop, payload, shellcode, x66, x6a, exploit, shell, execute, system, func
Rootkits	Java	2.50%	Rootkit, machine, virus, procedure, irp, tcp, stack, contents, flow
Memory Overflows	C/C++	2.50%	Overflow, crash, memory buffer, import, print, violation, access, software, pydbg
DLL Exploits	C/C++	1.25%	Figure, dll, module, analysis, memory, ldrloadll, calls, notice, breakpoints
Network Binding	C/C++	1.25%	Socket, return, port, error, case, int, sockaddr, tcp, http, protocol, buffer, bind

Bruteforcing	PHP	1.25%	Brute, force, website, password, form, access, user, php, pass
Total:	-	18.75%	-

Table 10. Malicious Source Code Topics in Reverse4You Forum

Similar to ExploitIN, some exploits in Reverse4You have shown recent popularity. Shellcode exploits and bruteforcers in particular have enjoyed a recent growth. While shellcode exploits, or exploits targeted at gaining root access over a target system, have been a consistently used tool over a long period of time, bruteforcing technology has garnered much recent interest. Such technology is often needed in environment which hackers need to try a multitude of username and password combinations. A recent example in which such a tool proved to be valuable can be seen with the FBI's purchase of a bruteforcing tool from hackers to help them unlock an iPhone related from the tragic events of the San Bernardino shooting (Constantin 2016). Such an example illustrates how tools acquired from hackers can prove to be valuable in a variety of contexts. Figure 7 highlights all of the malicious trends of assets in Reverse4You.

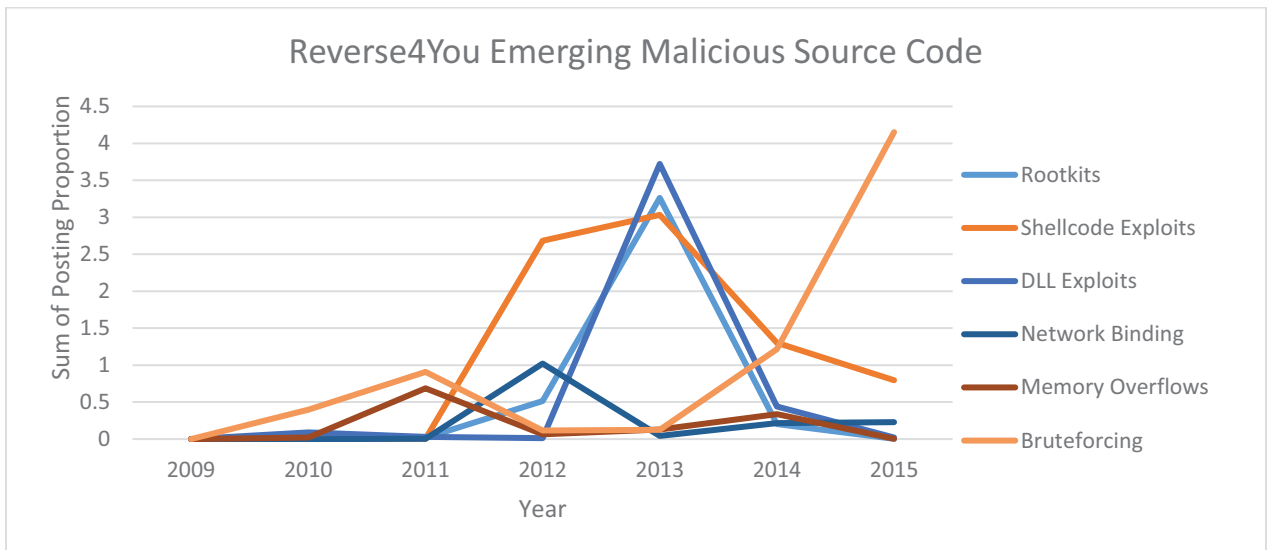


Figure 7. Emerging Malicious Source Code Topics for Reverse4You

2.5.4. Malicious and Emerging Attachment Topics

Although 80% of the topics modeled pertained to general topics, a variety of malevolent attachments was discovered. Table 11 presents some of the identified malicious topics.

Attachment Topics Label	% of topics	Keywords in topic(s)
DarkComet Remote Administration Tool (RAT)	4.00%	Remote, rat, darkcomet, website, title, users, capture, main, webcam, desktop, update, contain
Keylogging	4.00%	Keylogger, lttllogger, log, dll, injection, logfile, version, bypass, inject, automatic, key, folder
Password Cracking	3.00%	Password, check, rar, fake, program, download, victim, keygen, data, site, sniffer, software
Crypters	3.00%	Rar, crypter, coded, crypt, cryptor, decrypter, undetector, Trojan, polymorphic
Botnet/SYN Flood	3.00%	Attack, flood, icmp, udp, tcp, ddos, syn, attacker, attack, botnet, ping, control, execute
Password Stealers and Loggers	2.00%	Steal, version, upload, log, pass, password, have, key, username, site, logger, remove, secure, decrypt
Rootkits	2.00%	Hide, service, rootkit, windows, explorer, name, port, reg, space, start, registry, process, backdoor
Windows Process Injections	2.00%	Process, into, injection, pid, kill, windows, inject, list, running, dll, loaded, computer, close, system
Visual Studio Exploits	1.00%	Program, visual, Microsoft, include, studio, error, warning, ddos, buffer, flood, bind, shutdown, send
Browser Exploits	1.00%	Browser, database, default, address, exploit, exec, binary, install, checks, php, internet, websites
Binders	1.00%	Sub, binder, detect, build, download, pack, execution, bind, version, unpack, subs, extraction
Formgrabbing	1.00%	Data, system, application, form, content, post, username, account, script
Total:	27%	-

Table 11. Malicious Attachment Topics

Some of the malicious topics discovered were related to DarkComet Remote Administration Tools (RATs), comprising 4% of the topics. Keylogging makes up 3% of the topics, and password cracking tools about 3% of the topics. The DarkComet RAT is a popular Remote Administration Tool that allows an attacker to fully and remotely control a machine (e.g., power control, locking, file system access, etc.) and spy on a victim (e.g., keylogging, webcam captures, etc.). This kind of malware has been responsible for many of the violent escalations in the Syrian conflict in 2012 (McMillan 2012). Another popular topic, keylogging, while not as directly devastating as the DarkComet RAT, still has the potential to cause great damage. This software allows attackers to monitor victims' keystrokes and can potentially

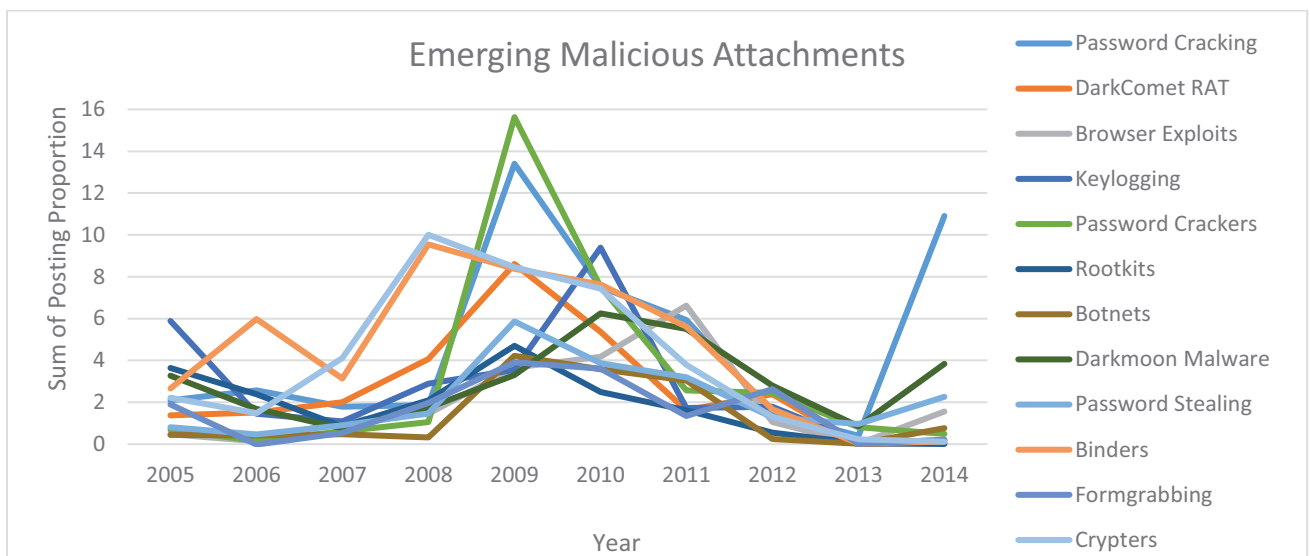


Figure 8. Emerging Malicious Attachment Topics

record sensitive information such as passwords, bank account information, and social security numbers. This information can then be used for other serious crimes such as identity theft or financial fraud.

Figure 8 illustrates that some emerging malicious topics are consistent with recent attacks. For example, Microsoft recently ended its support for Windows XP in early 2014.

Many attacks against this platform used DarkMoon Malware (Symantec 2014), one of the emerging topics. Password cracking tools show a significant spike in 2015. Weak and cracked passwords are often the easiest way to exploit a system (Kennedy et al. 2011). These types of tools may have been at the core of recent password exploitations against the Office of Personnel Management (OPM). Regardless of asset, 2012 marked the beginning of a significant downward trend in popularity or access appears for most attachments. Significant downtime in the forum OpenSC, the forum where the majority of the attachments come from, could be a cause for the decrease in attachments.

2.5.5. Malicious and Emerging Tutorial Topics

The majority of tutorial topics (90%) were benign, with a purpose of educating others on how to perform basic computing tasks. However, sets of malicious tutorials are still freely available for forum members to access, summarized in Table 12.

Tutorial Topics Label	% of topics	Keywords in topic(s)
DDoS	2.22%	Ddos, attacks, network, attack, security, malicious, bot, infected, large, traffic
Metasploit Tutorials	2.22%	Msfconsole, metasploit, tutorial, payload, exploit, shell, reversing, exploitation, vulnerabilities, tools
Shellcode Injections	2.22%	Shell, shellcode, memory, dll, exe, downloader, executable, injection, inject, remote
Carding Tutorials	1.11%	Sell, money, banks, bank, card, email, transfer, number, payment, paypal, info, account
Password Stealing	1.11%	Password, program, user, email, information, steal, account, victim, send, files
Crypter Tutorials	1.11%	Cryter, virus, scanner, stealer, binder, crypt, zip, istealer, tool, beta, generator
SQL Injections	1.11%	Page, php, sql, injection, html, action, request, cookie, error, select, web
Total:	11.10%	-

Table 12. Malicious Tutorial Topics

Overall, several types of malicious tutorials found are particularly interesting. First, carding tutorials educate hackers on illegal credit card manipulation and fraud. This type of crime is growing in notoriety in underground economies and is an emerging area of interest for both practitioners and researchers alike (Li et al. 2016). Second, crypter tutorials show how to develop tools that encrypt files, as well as how to build Ransomware. Finally, SQL injection tutorials develop a hackers' expertise in taking advantage of improperly coded websites to gain access to their underlying databases. Many popular retailers today are consistently hit with SQL injection attacks (Higgins 2014). Figure 9 visually illustrates the trends of these malicious topics.

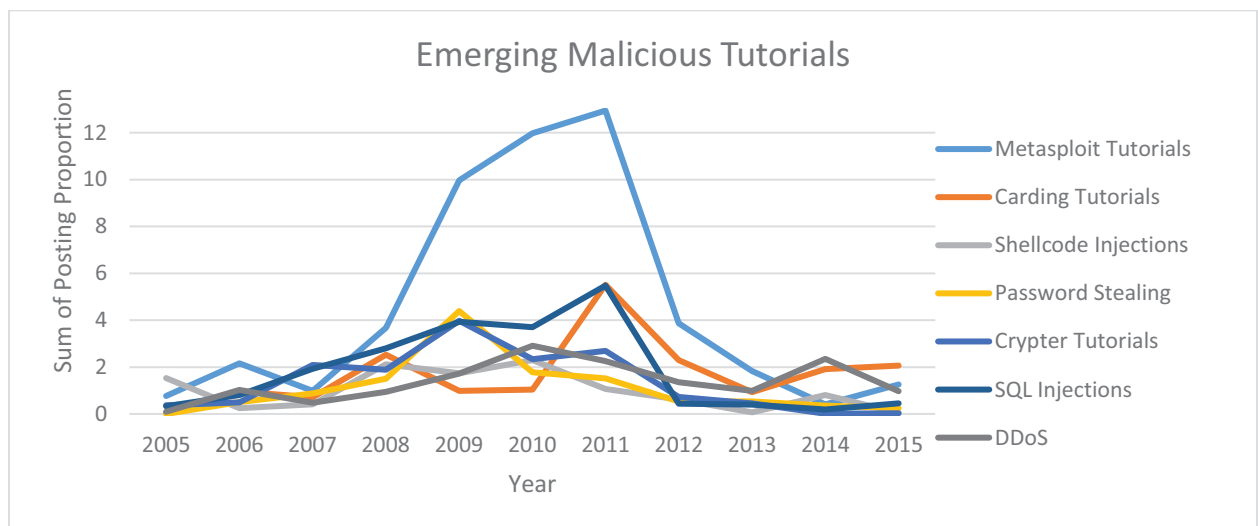


Figure 9. Emerging Malicious Tutorial Topics

Almost all of the malicious tutorials have leveled out in their popularity in recent years (right side of Figure 9). Among the discovered tutorials, some are consistent with major events. For example, Metasploit, a popular penetration testing framework, transitioned from Perl to Ruby from 2008-2011. During this time, Metasploit tutorials in forums grew, possibly due to hackers wanting knowledge about the new framework. Additionally, consistent with OpenSC

and ExploitIN source code, SQL injection topics increased between 2009 and 2011, a point in time when there were many exploitations against websites' databases (Wood 2009).

2.5.6. AZSecure Hacker Assets Portal

As mentioned in the research design, the development of the HAP can offer significant value to the larger cybersecurity community by allowing users to search, sort, browse, and download selected assets. For the first version of our portal, we loaded 15,576 source code snippets, 2,980 attachments, and 987 tutorials. Figure 10 illustrates the basic interface design for the HAP. For the purposes of illustration, we provide an example of how users can browse various categories of tutorials (identified from our LDA analysis), search and sort based on various post metadata (e.g., post date, author name, etc.) and access the raw tutorial. Tutorials are particularly valuable for cybersecurity education and CTI purposes as it allows users to understand exactly how hackers are executing their malicious tasks.

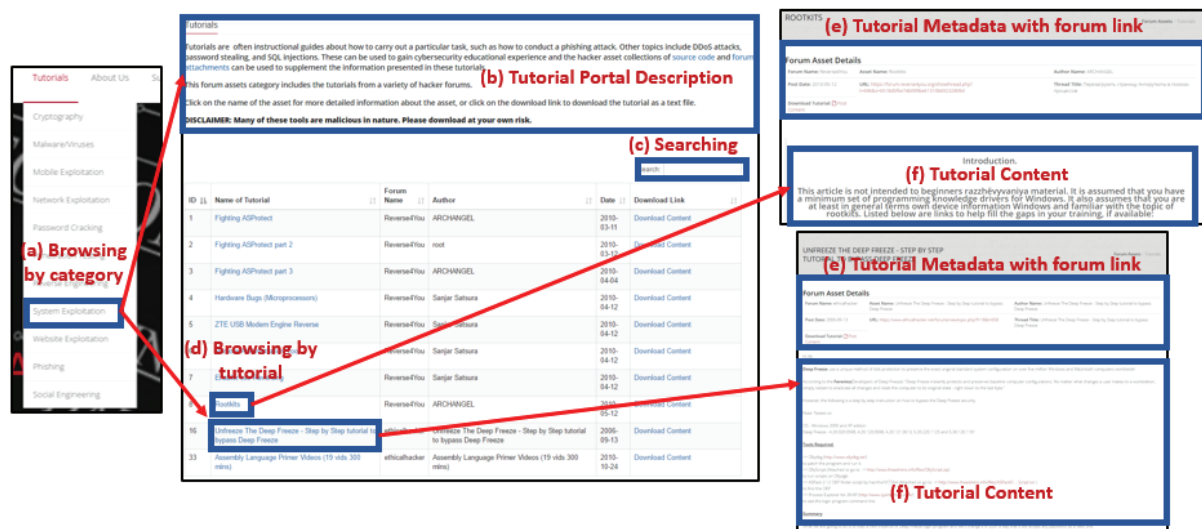


Figure 10. AZSecure Hacker Assets Portal

(a) Browsing tutorials by category, (b) description of tutorial portal, (c) searching for specific assets, (d) browsing by tutorial, (e) metadata associated with each tutorial and (f) tutorial content

In addition to the interface, we also created CTI dashboards (Figure 11) allowing users to identify trends of malicious assets and key threat actors for those assets (based on post

frequency). Built in Tableau, the dashboard dynamically updates based on users' selections of specific time points or hacker names. Assuming an organization knows their systems, the dashboard can provide a visual representation of asset trends and key inform future cyber-defenses. Given that CTI becomes more valuable with newer data, organizations interested in utilizing this framework are suggested to collect forums on a monthly or bi-monthly basis to ensure freshness of data.

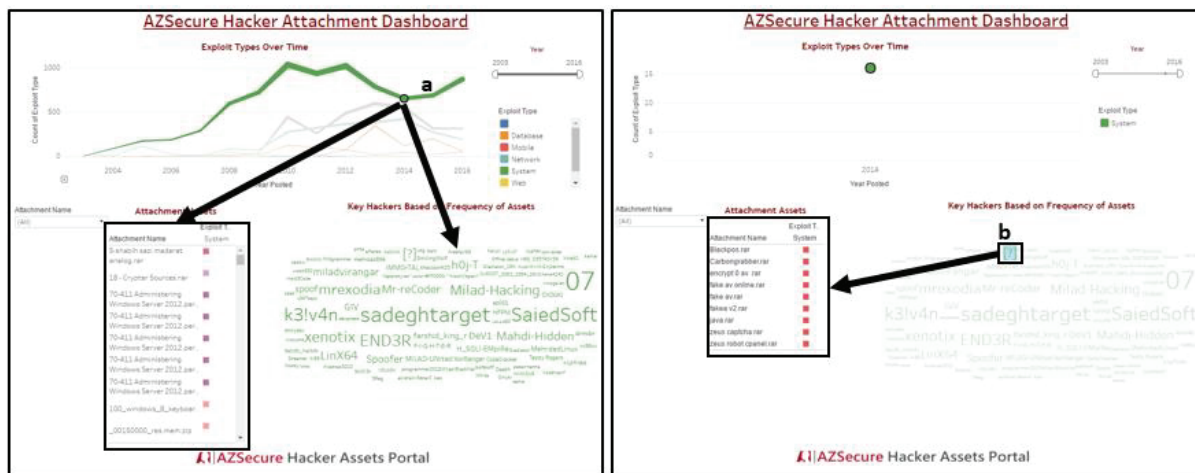


Figure 11. AZSecure Hacker Assets Portal Dashboard
(a) Filtering on 2014, when BlackPOS was posted, shows assets and threat actors at that time, (b) Filtering the actor who posted BlackPOS reveals that he posts other bank exploits (e.g., Zeus).

2.5.7. Social Network Analysis

The final part of our framework creates a bipartite social network between hackers and threads with source code assets for specific topics extracted by LDA. For illustration purposes, we create one bipartite network for crypter source code assets found in OpenSC. We represent the social network of crypter source code assets given their popularity in OpenSC and their recent notoriety for providing key technology for Ransomware. We also project the bipartite network into a monopartite network of hackers to calculate additional network measures. Figure 12 depicts both networks. A variety of topological and node level metrics (summarized in Table 13) are calculated to better understand the specific characteristics of the networks. Given that our goal is to identify key hackers for crypter source code assets, we focus the majority of our discussion on the monopartite hacker network.

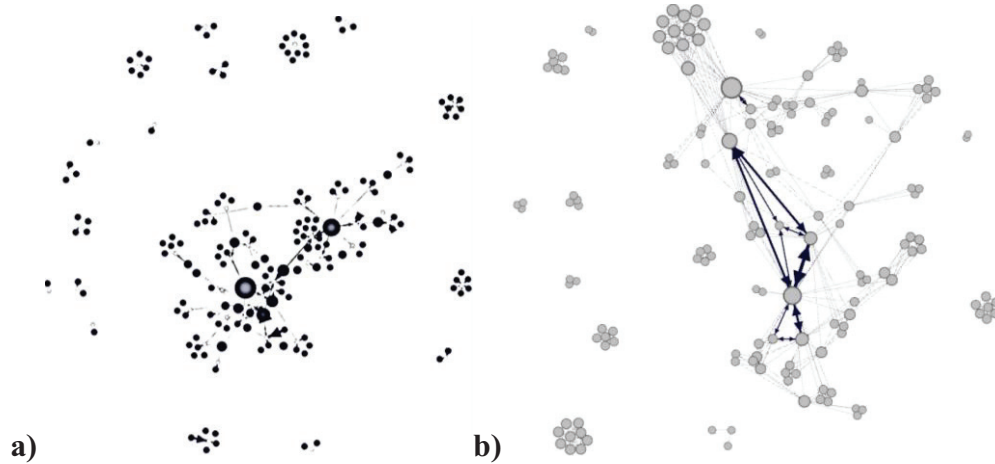


Figure 12. Crypter Bipartite and Monopartite Networks
(a) Original bipartite network of hackers connecting to threads containing crypter source code assets. Black nodes represent the hackers, white nodes represent the thread with a crypter asset. Size of the nodes represents degree centrality. (b) Monopartite projection of the hackers in the crypter network. Size of the nodes represents degree centrality.

Metric Category	Metric	Bipartite Network	Monopartite Hacker Network
Topological Metrics	Number of Nodes	207	156
	Number of Edges	207	938
	Network Diameter	1	6
	Graph density	0.005	0.039
	Connected components	18	14
	Size of giant component	159 (76.911%)	104 (66.67%)
	Average path length	1	3.106
Node Level Metrics	Minimum Degree	1	1
	Maximum Degree	14	68
	Average Degree	2	12.026

Table 13. Topological and Node Level Metrics for Crypter Bipartite and Monopartite Hacker Networks

Overall, there are 207 nodes in the bipartite network and 156 nodes in the monopartite hacker network. In the hacker network, we can see that the network diameter is 6, indicating that the network is relatively compact and that each hacker has to take a minimal amount of steps to reach another hacker in the network. This may be due to the fact that there is 14 connected components, with the size of the giant component comprising of 104 nodes, or 66.67% of the overall network. The giant component also contributes to a relatively low average path length of 3.106. However, even with the small network diameter and the low average path length, there is a small graph density (0.039), indicating that many of the hackers do not take advantage of the knowledge from all of the other hackers in the community, but instead from a select few. Such a discovery is consistent with prior literature (Holt et al. 2012), and is also represented with the large disparity in the degree distribution of the network. The majority of the hackers (103) have a degree less than the average of 12.026. Only a select few hackers have a high degree within the network. To better understand these hackers, we summarize the top 10 hackers for crypters based on their degree and betweenness centrality in

Table 14, both of which are strong indicators of key members within a network (Faust 1997).

We also detail each hackers' role in the forum, when they joined the forum, and the total number of posts they have made overall.

	Degree Centrality		Betweenness Centrality		Forum Status		
Hacker	Rank	Value	Rank	Value	Forum Role	Join Date	Total # of Forum Posts
K***r	1	68	1	4725.667	Admin	04/16/2006	2,008
m***5	2	54	2	4362.433	Senior Member	08/31/2008	3,053
c***6	3	42	3	3734.667	Member	03/14/2008	72
I***y	4	32	10	94.167	Senior Member	12/03/2005	151
c***i	5	30	4	1035.200	Senior Member	04/13/2009	1,913
c***n	6	30	8	578.667	Senior Member	12/20/2006	1,684
S***r	7	24	6	885.333	Senior Member	06/02/2008	1,535
R***s	8	24	7	706.267	Retired	03/07/2005	1,474
s***6	9	20	5	942.000	Senior Member	12/05/2007	1,440
d***x	10	20	9	470.000	Junior Member	01/01/2011	15

Table 14. Ranking and Forum Status for Key Hackers based on Degree and Betweenness Centrality

Several key insights can be taken from Table 14. First, the majority of the hackers in the top ten ranking are senior members within their community, have been part of the community for a long period of time, and contribute a large amount of forum posts. Such a reputation may help to drive the standing of these hackers as it pertains to the adoption and dissemination of crypter assets within the forum. Additionally, the total number of forum posts indicates that these hackers are active in the community as a whole, and not just within the crypter code network.

2.6. Conclusion and Future Directions

Although cybersecurity is a growing societal concern, much of the traditional cyber threat intelligence focuses on analyzing cyber-assets after they have already infected or compromised a system. Despite numerous technical challenges, there has been a push in recent years from practitioners and the information systems community alike to develop more proactive CTI. One way to help accomplish this is by understanding potential threats directly from hacker communities. This study contributes a novel framework to CTI by leveraging an automated and principled web, data, and text mining approach to collect and analyze vast amounts of hacker source code, tutorials, and attachments directly from large, international underground hacker communities. The framework allows us to identify many freely available, malicious assets in underground hacker forums such as crypters, keyloggers, SQL Injections, and password crackers, some of which may have been the root cause of recent breaches against organizations like OPM. We are also able to determine the key individuals behind these assets by utilizing social network analysis techniques and metrics. Our approach is generalizable to any hacker forum, irrespective of subforum structure.

One of the hallmarks of design research in information systems is the development of artifacts (e.g., systems) for practical utility (Nunamaker et al. 1990; Nunamaker et al. 2015; Nunamaker et al. 2017; Peffers et al. 2007; Prat et al. 2015). This research has practical implications for organizations aiming to improve their cybersecurity posture. Assuming an organization knows the systems they wish to protect, they can apply this framework to forums of their choosing to identify relevant hacker assets for their systems. Future work can expand this research in several directions. First, specific types of malware can be examined to identify the manner in which they disseminate and evolve over time, both between and within forums.

Second, additional analytical techniques such as sentiment analysis can be leveraged to supplement the social network analysis to better identify key hackers. Finally, novel machine learning techniques can be developed to identify key features of specific types of assets (e.g., Zeus code) to identify its key features and potentially predict how that malicious asset will evolve. All of these areas have the potential to further increase proactive cyber threat intelligence capabilities and prevent future cyber-attacks.

3. ESSAY II: LINKING HACKER COMMUNITY EXPLOITS TO KNOWN VULNERABILITIES: A DEEP STRUCTURED SEMANTIC MODEL APPROACH

3.1. Introduction

The widespread use of computing technologies has afforded modern society with unprecedented benefits. Industry, government, and academia use databases, communication networks, and other information systems (IS) to execute day-to-day operations. Unfortunately, malicious hackers often exploit these systems for cyberwarfare, hacktivism, espionage, or financial purposes, costing the global economy over \$450 billion annually (Graham 2017). To combat this dire issue, many organizations create, manage, and use knowledge about key hackers and emerging threats. This process, also referred to as Cyber Threat Intelligence (CTI), has emerged as a critical aspect of cybersecurity (Shackleford 2017).

To ensure the development of actionable CTI, organizations regularly assess the flaws of their systems and applications using vulnerability assessment tools such as Nessus, Qualys, or Burp Suite. Assessment results guide data collection from Intrusion Detection and Prevention Systems (IDS/IPS), and log files from databases, firewalls, and servers. Well-refined analytics procedures such as malware analysis, event correlation, and forensics then examine collected data to help derive the intelligence needed for CTI professionals to create cyber-defenses (Friedman 2015; Kime 2016; Shackleford 2016). Despite the maturity of prevailing CTI procedures, experts note the reliance on past events (e.g., log files) creates reactive intelligence (Bromiley, 2016). Consequently, major industry firms such as Ernst & Young (EY), have expressed that “cyber threats are increasing,” “businesses still aren’t doing enough to combat them,” and that “organizations need to take a more proactive approach to

cybersecurity” (EY, 2014). The SANS institute has further urged the use of “external threat intelligence sources to help alert the organization of threats it was not previously aware of” (Bromiley, 2016).

One data source strongly recommended by CTI experts to generate proactive CTI is the vast and evolving international online hacker community (Bromiley, 2016; Shackleford, 2016). The online hacker community is an appealing and novel CTI data source as it attracts and motivates millions of hackers from the US, Russia, Middle East, and China to share new hacking tools and knowledge. Today, the online hacker community comprises of four platforms: forums, Internet-Relay-Chat (IRC), DarkNet Markets (DNM), and carding shops (Benjamin et al. 2015). Among these, hacker forums allow users to freely share malicious tools designed to exploit vulnerabilities within popular systems and applications at many organizations. Exploits found in forums have been used in large-scale attacks, such as the BlackPOS malware for the Target breach (Kitten 2014). Figure 13 illustrates Windows Server 2008 and Android exploits available in hacker forums.



Figure 13. Example Forum Posts with Attached Android Exploits

Note: Forum posts with attached exploits designed for Windows Server 2008 Androids for free download. Exploit names generally describe the vulnerability they are designed for. Additionally, post content often includes information about the system or vulnerability exploit targets. Each post also has a date, valuable information when generating CTI.

Researchers and practitioners alike have found thousands of SQL injections, rootkits, crypters, and other malicious exploits within hacker forums (Samtani et al. 2017). However,

just discovering malicious tools cannot provide actionable intelligence; organizations must identify the exploits relevant to their vulnerabilities to formulate appropriate cyber-defenses (Shackleford 2016). Although many hacker exploit and vulnerability names within popular scanners share similar semantics (e.g., “Telnet Cracker” exploit and “Unencrypted Telnet Server” vulnerability), automatically creating accurate linkages is a non-trivial task. Hacker forum and vulnerability assessment data contain tens of thousands of unstructured, un-sanitized text records. Standard CTI analytics (e.g., malware analysis) cannot handle these unique characteristics. Moreover, behavioral and economic methodologies employed in other IS cybersecurity inquiries were not intended for such tasks. Thus, a robust computational Information Technology (IT) artifact is required.

One computational approach that can offer significant value in creating accurate exploit-vulnerability linkages is the Deep Structured Semantic Model (DSSM). Based upon deep learning, the DSSM employs sophisticated text representation processes to outperform baseline techniques in short text matching tasks (e.g., a query retrieving the most relevant document from a set) (Zhang et al. 2016). In this study, we adopt the design science paradigm to create a novel computational IT artifact, the Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM). Guided by the relevance and timeliness principles required for actionable CTI, the EV-DSSM extends the standard DSSM by encompassing key contextual cues to supplement exploit and vulnerability names. We also devise a novel severity score, the Device Vulnerability Severity Metric, based on the age and severity of linked exploits and vulnerabilities (respectively) to aid CTI professionals in prioritizing vulnerable devices for subsequent mitigation activities. Consistent with design science principles, we rigorously evaluate the EV-DSSM and its components against baseline approaches with a series of

benchmark experiments. We also demonstrate the EV-DSSM's and the DVSM's utility with two CTI case studies: openly accessible systems in major US hospitals and Supervisory Control and Data Acquisition (SCADA) systems. Apart from the extension in practical CTI applications, the EV-DSSM artifact presented opens up promising research avenues for IS researchers by providing novel design principles to guide the development of new computational IT artifacts.

The remainder of this essay is organized as follows. First, literature regarding past IS cybersecurity, hacker community research, vulnerability assessment, design science principles, and DSSM's is reviewed. We then summarize research gaps from the literature and pose research questions for study. Subsequently, we detail our research design and testbed. Key evaluation results and case study findings are then summarized. Finally, we discuss this study's relevance and contributions to IS literature, offer promising directions for future work, and conclude this study.

3.2. Literature Review

Our work draws on and is informed by three streams of literature: (1) hacker community research to gain knowledge on the exploits shared in these communities, (2) vulnerability assessment literature to discern methods to discover, assess, and rank vulnerabilities of systems, and (3) DSSM's to identify state-of-the-art methods to link hacker exploit and vulnerability names.

3.2.1. Hacker Community Research

Black hat hackers from the US, Russia, China, and the Middle East congregate on online platforms such as IRC channels, forums, DarkNets, and carding shops to exchange malicious tools, knowledge, and content relevant to their areas of expertise (Benjamin et al.

2015; Goel 2011; Samtani et al. 2016). Among these, forums are the predominant service for hackers to freely share malicious exploits (Hutchings and Holt 2015; Li et al. 2016; Samtani et al. 2015; Samtani et al. 2016; Sood and Enbody 2013; Zhao et al. 2016). Darknets are not cybersecurity focused, while IRC channels and carding shops platforms do not provide the technical mechanisms for hackers to post and access exploit tools (Ablon 2014; Benjamin et al. 2015; Marin et al. 2016; Nunes et al. 2016).

Previous studies have used Support Vector Machine (SVM), topic modelling, and interviews with subject matter experts to identify exploits in forums (Ablon et al. 2014; Hutchings and Holt 2014; Samtani et al. 2015; Samtani et al. 2016; Sood and Enbody 2013; Zhao et al. 2016). Analysis reveals that hackers share exploits such as botnets, email hacks, exploit kits, keyloggers, web exploits, remote administration tools (RAT's), bank exploits, and many others. As shown in Figure 13, each tool has a descriptive name indicating the vulnerability it is designed for. Forum posts also tend to describe what technology (e.g., system version) it targets (Nunes et al. 2016; Samtani et al. 2017). However, researchers note that simply identifying malicious exploits has minimal CTI value unless an organization knows the exploits targeting their specific vulnerabilities (Samtani et al. 2017; Shackleford 2016). Given the objective of this study is to create accurate links between hacker exploits and vulnerabilities, vulnerability assessment literature is reviewed to understand current approaches to discover and categorize vulnerabilities.

3.2.2. Vulnerability Assessment

Penetration testing literature defines a vulnerability as “a flaw within a system, application or service which allows an attacker to circumvent security controls and manipulate systems in ways the developer never intended” (Kennedy et al. 2011). As mentioned in the

introduction, organizations often use automated assessment tools such as Nessus or Qualys to identify and categorize their systems' flaws (Kennedy et al. 2011; Weidman 2014). Many modern tools can scan ports, find web application issues, discover unpatched technology, attempt logins, and many other functionalities. Each tool provides names and descriptions for every vulnerability it can test, valuable information when attempting to identify relevant exploits. Tools also specify the system versions susceptible to the vulnerability. Figure 14 depicts the contents of a typical vulnerability listing (name, synopsis, description, severity score, and description of vulnerable systems) from Nessus, a gold-standard assessment tool used in the industry (Weidman 2014).

Cisco IOS IPS Denial of Service Vulnerability - Cisco Systems **a**

Synopsis **b**
The remote device is missing a vendor-supplied security patch.

Description **c**
The Cisco IOS Intrusion Prevention System (IPS) feature contains a vulnerability in the processing of certain IPS signatures that use the SERVICE.DNS engine. This vulnerability may cause a router to crash or hang, resulting in a denial of service condition.

Risk Information
Risk Factor: High
CVSS Base Score: 7.8 **d**
CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:N/I:N/A:C

Reference Information:
CVE: CVE-2008-2739
OSVDB: 48711
BID: 31364 **e**

Cisco IOS IPS 'SERVICE.DNS' Remote Denial of Service Vulnerability

Bugtraq ID: 31364
Class: Failure to Handle Exceptional Conditions
CVE: CVE-2008-2739
Remote: Yes
Local: No
Published: Sep 24 2008 12:00AM
Updated: Sep 24 2008 08:19PM
Credit: The discoverer of this issue is not known; this issue was disclosed by Cisco.
vulnerable: Cisco IOS 12.4YA
Cisco IOS 12.4XZ
Cisco IOS 12.4XY
Cisco IOS 12.4XW
Cisco IOS 12.4XV
Cisco IOS 12.4XT

Figure 14. Sample Vulnerability Descriptions

Each vulnerability has (a) a short title, (b) one sentence synopsis, (c) a longer description of the vulnerability, (d) a vulnerability severity score, and (e) system versions vulnerable to exploitation

Tools also provide a severity score for each vulnerability. Scores range from 0.0-10.0. Scores are often segmented into 'Critical,' 'High,' 'Medium,' and 'Low' thresholds. Each vulnerability's score is calculated based on the Common Vulnerability Scoring System (CVSS) (FIRST 2017). CVSS is an open industry standard aiming to standardize how vulnerability information is shared and interpreted across the cybersecurity community (FIRST 2017). CVSS includes vulnerability type (e.g., network), age, ramifications if vulnerability is

exploited, and known exploits from published databases into its calculations (FIRST 2017).

Table 15 summarizes CVSS score ranges, rankings, and examples.

Severity Ranking	CVSS Range	Examples
Critical	9.0 – 10.0	Unsupported operating system, PHP Unsupported Version Detection, OpenSSL Unsupported, Windows out-of-date
High	7.0 – 8.9	SQL Injections, OpenSSH vulnerabilities, Buffer Overflows, Linux Chunk Handling Remote DoS
Medium	4.0 – 6.9	DoS, XSS, Browseable Web Directories, OpenPAM DoS, Unencrypted Telnet Server, Dropbear SSH vulnerabilities
Low	0.1 – 3.9	Cleartext submission of credentials, authentication without HTTPS, SSL Cipher issues

Table 15. CVSS Score Ranges, Rankings, and Examples

CVSS scores have two key limitations. First, there is no standard approach to aggregate severity scores for devices with multiple vulnerabilities (FIRST 2017). The lack of a standard approach to aggregate severity scores in these devices causes organizations to struggle prioritizing their most vulnerable systems. Second, CVSS scores do not account for hacker community exploits into its calculation. Part of the reason for this exclusion is the difficulty in creating mappings between hacker exploits and known vulnerabilities.

3.2.3. Deep Structured Semantic Model (DSSM)

Design science principles suggest searching a space of possible solutions (Hevner et al. 2004). In the context of computational artifacts, this often involves selecting an algorithm or approach based on key aspects of the researcher’s intended domain. Thus far, our review indicates that hacker exploit names detail the type of vulnerability they are designed for. Similarly, vulnerability names provide cues as to the types of tools they can be exploited with. For example, the vulnerability name of “*BACnet buffer overflow*” has high relevancy to the “*buffer_overflow_exploit*” exploit name. One promising approach that can leverage these characteristics to create exploit-vulnerability linkages is the deep learning based Deep

Structured Semantic Model, also known as Deep Structured Similarity Model (DSSM). Introduced by Huang et al. (2013), DSSM's are a form of multi-view learning that computes semantic similarity (i.e., similarity based on meaning) between two short text inputs. Since its inception, DSSM's have been used primarily for ad-hoc information retrieval (IR), where a query aims to retrieve the most relevant document from a set (Zhang et al. 2016). DSSM's have shown significant statistical improvement over traditional baselines of Term Frequency-Inverse Document Frequency (TF-IDF), Latent Semantic Analysis (LSA), and Best Matching 25 (BM25) (Zhang et al. 2016).

The first DSSM component is a word hashing technique to represent text. Unlike traditional approaches that compare two text phrases directly, the DSSM extracts letter n-grams, usually letter trigrams, from each phrase (Huang et al. 2013). For example, the phrase *“buffer overflow”* would be hashed to *#bu, buf, uff, ffe, fer, er#, #ov, ove, ver, rfl, flo, low, ow#*. Processing text in this manner significantly increases DSSM's robustness to noise and word variations, a common concern when analyzing social media text (Zeng et al. 2010). Additionally, it captures finer linguistic cues such as roots and morphs missed by competing approaches.

Each word hashed phrase is inputted into its own multi-layer feed-forward deep neural network (DNN). DNN's and its variations have achieved unprecedented performance in applications such as image processing, language translation, and many others (LeCun et al. 2015). In the DSSM, the DNN takes the high-dimensional word hashed phrase as input and outputs a low-dimensional embedding in latent semantic space. To achieve this, each layer in the DNN receives the output of the previous layer and reduces its dimensionality by applying a non-linear transformation (i.e., activation function). For example, the first DNN layer would

reduce the high dimensional word hashed input to 30k dimensions. The second layer reduces the 30k dimensions to 300, followed by the third layer reducing the 300 to 128 dimensions, and so on. This procedure offers two key benefits. First, the DNN will discover important semantic structures found in the input text that are missed by competing approaches. Second, comparing low-dimensional embeddings is more computationally efficient than directly comparing high-dimensional raw phrases. For example, comparing embeddings that are 128 dimensions requires significantly less computation power compared to comparing embeddings that are 30k dimensions.

After creating embeddings, cosine similarity calculates the distance between two embeddings. The similarity score is passed through a softmax function to calculate the conditional probability (i.e., $P(D|Q)$) for a document (D) – query (Q) pair. Phrases with similar embeddings appear closer to each other in semantic space, while dissimilar embeddings are further away. The document title with the highest conditional probability with the query is labeled as the most relevant. Figure 15 depicts a DSSM and its components (word hashing, multi-layer non-linear projections, cosine similarity, and softmax function) in an IR task.

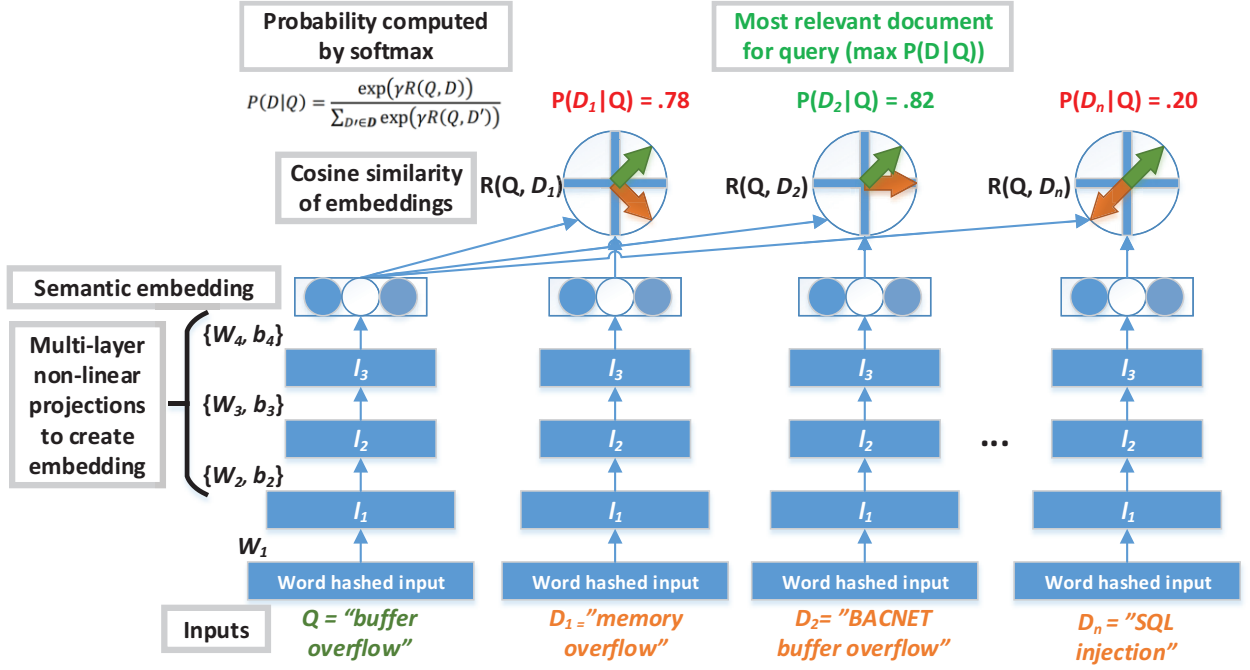


Figure 15. Deep Structured Semantic Model (DSSM) Architecture (Adapted from Huang et al. 2013)

Both the query and document title inputs are word hashed. Each word hashed phrase is projected through multiple non-linear activation functions (usually tanh) of its DNN to create embeddings in semantic space. Cosine similarity computes the relevance between query and document embeddings. The conditional probability the document belongs to the query is calculated with the softmax function. The softmax function takes in the cosine similarity of Q-D and divides it by the relevance of other documents. The document with the highest score is the most relevant to the query. In this example, the “BACNET buffer overflow” document is most relevant to the “buffer overflow” query.

DSSM’s have been used to search news articles (Guo et al. 2016; Pang et al. 2016), retrieve social media posts (Jaech et al. 2017; Song et al. 2016), and rank web pages (Huang et al. 2013; Shen et al. 2014; Song et al. 2016; Mitra et al. 2016). Studies have adjusted the standard DSSM by substituting the feed forward network with a convolutional neural network (CNN) to capture word sequences from phrases (Gao et al. 2014; Shen et al. 2014) or changed the model structure to better fit their context (Jaech et al. 2017; Mitra et al. 2016; Pang et al. 2016). Irrespective of application, all studies note that DSSM’s perform best when using only the query and the document title for matching, not the entire document (Zhang et al. 2016).

However, they also mention that utilizing only query and document titles omits valuable selected contextual information from the document contents (e.g., key phrases) that can develop more accurate linkages (Zhang et al. 2016). Table 16 summarizes the tasks, data sources, and model types in selected DSSM literature.

Year	Authors	Task	Data Source	Model Type	Evaluation Metric
2017	Jaech et al.	Query-document matching	1.6M query-doc from social media	DRMM	NDCG
2016	Song et al.	User's news clicks	26.5M training pairs	T-DSSM	NDCG
2016	Guo et al.	Ad-hoc information retrieval	Robust04 news collection, ClueWeb09-Cat-B	DRMM	NDCG
2016	Pang et al.	Ad-hoc information retrieval	Robust04 news collection	MatchPyramid (DRMM)	NDCG
2016	Mitra et al.	Ad-hoc information retrieval	199,753 queries, 998,765 documents	Duet (DRMM and DSSM)	NDCG
2015	Ye et al.	Ad-hoc information retrieval	Query logs from search engine	DSSM	NDCG
2014	Shen et al.	Queries for web documents	12,071 queries, 65 documents	C-DSSM	NDCG
2014	Shen et al.	Web document ranking	12,071 queries, 65 documents	CLSM	NDCG
2014	Gao et al.	Identifying similar documents	18 million user click instances	C-DSSM	NDCG
2013	Huang et al.	Web document ranking	16,510 queries, 15 documents	DSSM	NDCG

Table 16. Selected Literature Using DSSM's

Note: Note: DRMM = Deep Relevance Matching Model; T-DSSM = Temporal-Deep Semantic Similarity model; C-DSSM=Convolutional Deep Semantic Similarity Model; CLSM=Convolutional Latent Semantic Model; NDCG=Normalized Discounted Cumulative Gain

All DSSM related studies use the Normalized Discounted Cumulative Gain (NDCG) metric to evaluate their selected DSSM approach (Zhang et al. 2016). NDCG is a popular metric for evaluating an algorithm's ability to identify the most relevant document order for a set of documents for a given query. NDCG compares how accurately an algorithm retrieves documents for a query compared to the ground truth. The closer the algorithm's rank order is to the ground truth order, the higher the NDCG score. DSSM studies calculate and report the NDCG score at the 1, 3, and 10 thresholds. NDCG@3 would compare the algorithms top three ranked documents against the ground truth, NDCG@10 would be on the top 10, and so on. Figure 16 illustrates the NDCG metric formulas and an example.

➤ NDCG evaluates the overall relevance level of a specific document order.

- **Relevance:** Each document d has a relevance score rel_d
- **Position:** The contribution at rank r ($r > 1$) is discounted by $\log^{-1}(r)$
 - Discount penalizes documents that do not appear in the order of the ground truth

Discounted Cumulative Gain (DCG)
Weighted sum of rel_d w. r. t. rank r

$$DCG(r) = rel_{d_{r1}} + \sum_{i=2}^p \frac{rel_{d_{ri}}}{\log(i)}$$

Normalized DCG (NDCG)
 $NDCG_i(r) = \frac{DCG_i(r)}{DCG_{GT}(r)}$

Rank (r)	Discount factor	Ground Truth (GT)		Ranking Function, (RF1)	
		Order	rel _d	Order	rel _d
1	1	d ₄	2	d ₃	2
2	$\log^{-1}(2)$	d ₃	2	d ₂	1
3	$\log^{-1}(3)$	d ₂	1	d ₄	2
4	$\log^{-1}(4)$	d ₁	0	d ₁	0
DCG(4)		DCG _{GT} =4.63		DCG _{RF1} =4.26	
NDCG(4)		NDCG _{GT} =1.00		NDCG _{RF1} =0.92	

Figure 16. Illustration of the NDCG Metric

3.3. Research Gaps and Questions

Our literature review revealed several key research gaps. First, existing IS literature has made extensive contributions in various areas of cybersecurity literature, but cybersecurity analytics and hacker community studies are still relatively unexplored. Second, existing hacker forum studies in related domains have identified exploits in forums, but we are unaware of any work attempting to link the tools to known vulnerabilities. Third, vulnerability assessment literature indicates that current practices do not account for hacker exploit information in the CVSS standard, nor calculate device level severity scores for systems that have multiple vulnerabilities. Finally, DSSM literature utilizes only query and document titles to create

linkages, omitting valuable contextual information that can develop accurate linkages. These gaps motivate the following research questions for study:

- How can DSSM's link hacker forum exploit names to vulnerability names by considering cybersecurity related contextual information?
- What types of vulnerabilities do hacker exploits target?
- How can device level severity scores be calculated that incorporate vulnerability and hacker exploit information to facilitate CTI?

3.4. Research Design and Testbed

3.4.1. Data Collection: Hacker Forum Tool Extraction and Vulnerability Assessment Data

The data collection component of the design aims to collect a large set of hacker exploits and compile a list of vulnerability names. To collect hacker exploits, we selected three (two English, one Arabic) large and long-standing forums well-known in the hacker community for containing a variety of malicious tools. After forum selection, a web crawler routed through the Tor network collects and parses all exploit, post date, and post content into a relational database. These procedures are consistent with prior hacker community literature (Benjamin et al. 2015; Li and Chen 2014; Samtani et al. 2015; Samtani et al. 2016). Collection efforts resulted in a testbed of 14,641 exploits that include Zeus malware, botnets, various web exploits, buffer overflows, SSH exploits, and DDoS attacks. Table 17 summarizes each forum. Forum names are anonymized to protect us from breaches by hackers in these communities.

Forum	Language	Date Range	Number of Posts	Number of Members	Number of Exploits
O****C	English	2/7/2005 – 12/15/2016	124,993	6,796	2,349
A****e	Arabic	5/30/2003 – 12/15/2016	34,247	6,406	10,086
T****u	English	6/10/2006 – 12/15/2016	40,666	2,539	2,206
Total:	-	2/7/2005- 12/15/2016	199,906	15,741	14,641

Table 17. Summary of Hacker Forum Tool Collection

After collecting hacker exploits, we compile a comprehensive list of vulnerability names, their descriptions, and severity scores. Securityfocus.com is a trusted INFOSEC resource providing vulnerability information for many tools such as Nessus, Qualys, and Burp (First 2017). We crawl Security Focus to extract all vulnerability listings. The collection contains 64,323 listings in the ‘Critical’, ‘High’, ‘Medium’, and ‘Low’ levels. Table 18 summarizes the number of vulnerability listings in each threshold.

Severity Ranking	CVSS Range	Number of Vulnerability Listings
Critical	9.0 – 10.0	8,355
High	7.0 – 8.9	24,098
Medium	4.0 – 6.9	28,707
Low	0.1 – 3.9	3,163
Total:	–	64,323

Table 18. Summary of Security Focus Vulnerability Information Collection

3.4.2. Exploit Vulnerability Linkages: Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM)

While the standard DSSM can be applied to find the most relevant exploit for a particular vulnerability by just using their names, doing so omits valuable contextual information from hacker post content and vulnerability descriptions. As shown in Figure 13, hacker forum posts containing exploits often specify the technology they are designed for. Similarly, Figure 14 depicted how vulnerability descriptions provide lists of systems susceptible to that vulnerability. From a CTI professional’s point of view, a hacker exploit becomes significantly more relevant if it targets a specific technology on their network (Friedman 2015). For example, if a buffer overflow vulnerability is in a Windows XP system on an organization’s network, a buffer overflow hacker exploit designed specifically for the Windows XP system would be significantly more relevant than a general buffer overflow exploit. These important domain characteristics and perspectives motivate the extension of the

standard DSSM to a novel, context enriched model entitled the EV-DSSM (Exploit-Vulnerability DSSM). The EV-DSSM IT artifact leverages valuable contextual cues from the hacker forum post content and vulnerability descriptions to supplement hacker forum exploit and vulnerability names.

The EV-DSSM operates as follows. Prior to word hashing an exploit-vulnerability pair, all vulnerable version names are extracted from a vulnerability description and cross-referenced with the hacker exploit post. If a version name appears in the hacker forum post, then the version is appended to both the vulnerability and exploit name. The pair then follows the standard DSSM procedure of letter trigram word hashing and DNN processing to create semantic embeddings for comparison. Intuitively, appending the susceptible version information to both exploit and vulnerability names will increase the similarity between their semantic embeddings. Figure 17 visually depicts an example of the EV-DSSM.

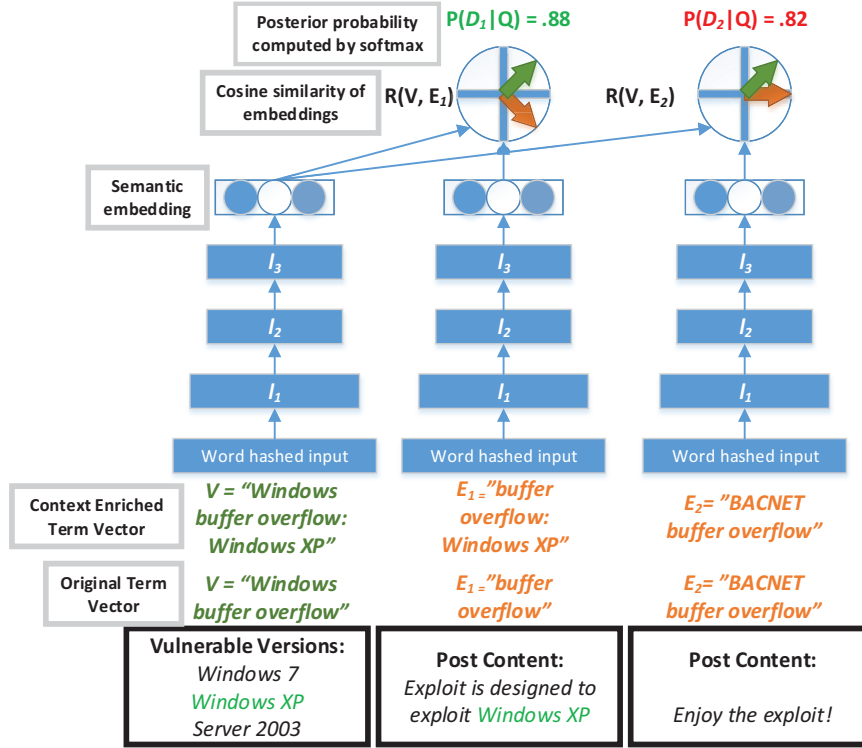


Figure 17. Exploit-Vulnerability-Deep Structured Semantic Model (EV-DSSM) Architecture

The EV-DSSM extracts the vulnerable versions from the vulnerability description and cross-references them with hacker forum post content. If a version appears in the post, (e.g., center), the version is appended to the vulnerability and exploit names before hashing. Intuitively, this increases their similarity and creates more accurate linkages.

Pre-processing and training procedures suggested by prior DSSM work are closely followed (Zhang et al. 2016). All text is translated to English, tokenized based on whitespace, and lowercased. Content is not stemmed or lemmatized. Pre-processing text in this manner reduces the chances for creating inaccurate linkages by normalizing many irregularities such as capitalization and language differences (Huang et al. 2013). The EV-DSSM is then trained with a gold-standard of 4,000 human labeled exploit-vulnerabilities pairs. We adopt the standard letter trigrams word hashing approach, the tanh activation function for the DNN, and mini-batch based stochastic gradient decent (SGD) for training (Zhang et al. 2016). All EV-

DSSM operations were implemented on a single machine with an NVIDIA GEFORCE GTX 1070X GPU, 32 GB of RAM, and 1 TB of disk space.

The EV-DSSM matching procedure is denoted in Algorithm 1. The input is two sets: a set of exploit names and their associated hacker forum post content, and a set of vulnerability names with their respective vulnerability descriptions. A distinct tuple for each possible exploit-vulnerability combination is created. In each tuple, we extract the vulnerable versions from the vulnerability description and cross-reference them against the hacker forum exploit posting. If the vulnerable version name appears in the posting, then the version name is appended to the hacker exploit and vulnerability names. The tuple is then word hashed, and passed through the DNN to create low dimensional embeddings in semantic space. Embeddings are compared with cosine similarity and passed through the softmax function to compute the conditional probability (i.e., $P(E|V)$). This process is repeated for all tuples. After computation, the exploit with the highest conditional probability with a vulnerability is labeled as the most relevant.

ALGORITHM 1. Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM)

Input: exploit list $E = \{e_1, e_2, \dots, e_n\}$ and vuln. list $V = \{v_1, v_2, \dots, v_n\}$

Output: a list of tuples, $[(e_{k1}, v_1), (e_{k2}, v_2), \dots]$ where $e_{ki} \in E, v_i \in V$ ($i = 1, 2, \dots, n$)

Procedure

for $v_i \in V$ ($i = 1, 2, \dots, n$):

for $e_j \in E$ ($j = 1, 2, \dots, n$):

if $v_i.vuln_ver$ in $e_j.post_content$: // vulnerable version matching

$v_i.title, e_j.title = \text{concat}(v_i.title, v_i.vuln_ver), \text{concat}(e_j.title, v_i.vuln_ver)$

else:

 do nothing

$v_i.\text{hashed_vulntitle}, e_j.\text{hashed_explttitle} = \text{word_hash}(v_i.title), \text{word_hash}(e_j.title)$ // word hashing

foreach DSSM layer: // create semantic embedding for vulnerability title with multi-layer DNN projections

$\text{sem_embed_}v_i = \text{tanh_activation}(v_i.\text{hashed_vulntitle})$

foreach DSSM layer: // create semantic embedding for exploit title with multi-layer DNN projections

```

    sem_embed_ej = tanh_activation(ej.hashed_expltitle)
    sim_ij = cos_sim(sem_embed_vi, sem_embed_ej) // compute cosine similarity
    cond_prob_ij = softmax(sim_ij) //conditional probability computed by softmax
  end for
  k_i = argmax_j(cond_prob_ij) // identify most relevant exploit based on max conditional
  probability
  output.append((e_ki, v_i))

```

3.4.3. Exploit-Vulnerability Linkages: Benchmark Evaluations

One of the hallmarks of IS design science research is the rigorous evaluation of proposed IT artifacts (Gregor and Hevner 2013; Hevner et al. 2004; Peffers et al. 2007; Rai 2017). In this study, the EV-DSSM is evaluated with two sets of experiments. Table 19 describes each set.

Experiment	Justification	Methods	Description	Evaluation
Baseline	Standard benchmarks (Huang et al. 2013; Zhang et al. 2016)	DSSM	Standard DSSM with no exploit or vulnerability content extraction	NDCG@1, 3, and 10, paired t-tests to evaluate statistically significant differences
		LSA	Distributional semantics model that calculates semantic similarity between phrases based on word location, not term frequency	
		BM25	State-of-the-art probabilistic bag-of-words retrieval function based on term frequencies	
		TF-IDF	Both names are weighted based on term frequency	
Word Hashing	Identify ideal input text representation for algorithm. Exploit and vulnerability names often contain content (e.g., version number) not in other DSSM applications	Letter trigrams	Standard word hashing approach	
		Letter bigrams	Extracts letter bigrams (e.g., <i>buffer</i> → # <i>b</i> , <i>bu</i> , <i>ff</i> , <i>fe</i> , <i>er</i> , <i>r</i> #)	
		Word n-grams	Extracts word n-grams (e.g., <i>buffer overflow</i> → <i>buffer</i> , <i>overflow</i>)	

Table 19. Summary of Benchmark EV-DSSM Evaluations

The first set of experiments benchmarks the EV-DSSM against common baselines: the standard DSSM, LSA, BM25, and TF-IDF. As with prior literature, the NDCG metric evaluates all approaches (Zhang et al. 2016). We evaluate at NDCG@1, 3, and 10 levels. Paired t-tests calculate statistically significant differences in performance. Differences are significant if $p < 0.05$. Benchmarking in this fashion is commonly accepted practice in DSSM literature (Zhang et al. 2016).

The second set of experiments evaluates the word hashing component of the DSSM, a key step in representing the input text for the subsequent DNN processing. Prior literature often extracts letter trigrams from the text for input into the neural network (Zhang et al. 2016). However, all prior studies have operated the DSSM in natural language contexts. Vulnerability and hacker exploit names contain content not in natural language (e.g., version numbers). Thus, we aim to identify the ideal word hashing configuration for the EV-DSSM to aid future DSSM research operating in non-natural language contexts. To execute these experiments, both the EV-DSSM and the DSSM are trained with the standard letter trigram approach as well as letter bigrams and word n-grams. As with the first set of experiments, performance is evaluated with NDCG@1, 3, and 10. Paired t-tests calculate statistically significant differences.

3.4.4. Exploit-Vulnerability Linkages: Device Vulnerability Severity Metric (DVSM)

Calculation

The EV-DSSM offers a systematic approach to discover the most relevant hacker exploit for a vulnerability. Coupling key domain characteristics with EV-DSSM's output can create holistic and valuable CTI. First, devices are often afflicted with multiple vulnerabilities, each with their own severity score. However, we are currently unaware of any standard approach to aggregate vulnerability severities in devices with multiple vulnerabilities. Second,

each hacker exploit has a date when it was posted in the forum. Knowing when an exploit was posted is valuable information from a CTI perspective. In general, newer exploits, such as zero-days, are significantly more valuable than older ones. As an exploit ages, knowledge about its existence, how it operates, and the suitable cyber-defenses required is quickly disseminated across the cybersecurity community (FIRST 2017). As a result, exploits exponentially lose value over time (FIRST 2017).

Since the EV-DSSM can determine the most relevant exploit for a vulnerability and that all vulnerabilities in a system can be detected with scanning tools, we develop a novel Device Vulnerability Severity Metric (DVSM) encompassing the number of vulnerabilities in a device, each vulnerability's severity, and the hacker exploit age for each vulnerability. This severity metric can enable more efficient mitigation and investigation activities for CTI professionals by allowing them to prioritize key devices on their networks. Table 20 summarizes key metric features.

Feature Category	Feature	Justification for Inclusion	References
Vulnerability	Vulnerability severity (CVSS score, 0.0-10.0)	A higher severity score indicates more severe consequences if device is compromised (e.g., total system failure)	Rouse 2017; FIRST, 2017; Weidman 2014; Kennedy et al. 2011
	Number of device vulnerabilities	Devices with more vulnerabilities have a higher exploit susceptibility	
Hacker Exploit	Number of hacker exploits targeting vulnerabilities	More hacker exploits targeting a vulnerability increases the probability of the devices being harmed	Friedman 2015; Robertson et al. 2017
	Age of hacker exploits (i.e., forum post date)	Newer exploits are more valuable for CTI as there has been less time to formulate defenses. Older exploits have less CTI value	Shackleford 2016; Kime 2016

Table 20. Features for Device Vulnerability Severity Metric

Formally, the Device Vulnerability Severity Metric (DVSM) is denoted as:

$$D = \sum_{j=0}^J \left(\frac{s_j}{\log(d_j + 2)} \right)$$

Where:

D is the overall device severity score

J is the number of vulnerabilities in a system

s_j represents the severity of a specific vulnerability within the device

d_j is the # of days since the most relevant exploit for the vulnerability s_j was posted (end date 6/1/2017), creating a decaying effect of the inverse log function

The most relevant exploit for a particular vulnerability is determined by the EV-DSSM.

A vulnerability's severity score is divided by the log of the number of days elapsed since the most relevant exploit for that vulnerability was posted (a decaying function). Intuitively, severities receive a higher weighting in the metric if its most relevant exploit is newer. Although there are a variety of functions that can be applied on the age, the inverse log best captures the exponential loss of value detailed in prior CTI literature (FIRST 2017). All vulnerability score and hacker exploit age pairs for a device are summed to create the overall device severity score. A device's overall score is higher if it has more severe vulnerabilities or newer exploits for vulnerabilities. Conversely, a device's severity score is lower if it has a smaller number of less severe vulnerabilities and older exploits for vulnerabilities. We note that the DVSM formulation is generic in nature: researchers and organizations with more specialized needs can adjust the base formula to suit their application.

3.4.5. Case Studies: US Hospitals and SCADA Devices

As the last part of our framework, we demonstrate the utility and value of our research with two case studies: networks of the top eight US hospitals and SCADA devices deployed worldwide from major vendors such as Rockwell Automation, Siemens, and Schneider Electric. Both hospitals and SCADA devices have become prime targets for hackers in recent

years (Ayala 2016; Samtani et al. 2016). Hackers infiltrate hospitals with hopes of exploiting databases to extract sensitive medical records to sell on DarkNet Marketplaces (Ayala 2016). SCADA devices are ideal for exploitation as they control important components of modern infrastructure including power plants, sewage, transportation, factory automation, and many others. Harming them can severely cripple day-to-day operations in society. Our framework provides stakeholders in either context to prioritize devices for subsequent investigation and threat mitigation activities.

The process for each case study is as follows. For hospitals, we identify hacker exploits for vulnerabilities on the externally facing networks of the top eight US hospitals as ranked by the 2017 US News and World Report: (1) Mayo Clinic, (2) Cleveland Clinic, (3) Massachusetts General Hospital, (4) Johns Hopkins Hospital, (5) UCLA Medical Center, (6) New York Presbyterian, (7) UCSF Medical Center, and (8) Northwestern Memorial. After identifying hospitals, Shodan, the well-known search engine that discovers publicly accessible Internet of Things (IoT), finds all devices available on each hospital's IP range. Subsequently, Nessus, a state-of-the-art vulnerability assessment tool used in many industry assessments, discovers the vulnerabilities of each device. To ensure we did not adversely affect any device, all port scanning and payload dropping activities are avoided. Such practices are consistent with past literature (Samtani et al. 2018; El et al. 2017; Williams et al. 2017; Samtani et al. 2016; McMahon et al. 2017). Following the assessment, the EV-DSSM determines the most relevant hacker exploit for each vulnerability. The DVSM then ranks the vulnerable devices for each hospital. The SCADA case study identifies the vulnerabilities and their most relevant hacker exploits on SCADA devices publicly available on Shodan. Unlike the hospital case study, a set of ICS related keywords retrieves 20,641 SCADA devices from Shodan. Nessus scans these

devices, the EV-DSSM links exploits and vulnerabilities, and the DVSM ranks them accordingly.

The process of reconnaissance with Shodan, vulnerability identification using Nessus, linking with EV-DSSM, and prioritization with DVSM mimics the series of steps commonly used by hackers (Kennedy et al. 2011; Weidman 2014). However, hackers generally execute one additional step: exploitation. Unlike hackers, we cannot drop exploits against organizations, even to validate the accuracy of the EV-DSSM linkages, without significant ethical and legal consequences. Moreover, many organizations suitable for partnership are often hesitant to disclosing the possible exploits against them, preferring instead to keep this knowledge classified. Given these constraints, we stop our process at the linkage and ranking phase of publicly accessible devices and save the process of validating hacker exploits in a safe and non-intrusive manner for future work.

3.5. Results and Discussion

3.5.1. EV-DSSM Baseline Evaluation

As mentioned in the research design, the EV-DSSM is benchmarked against the standard DSSM, LSA, TF-IDF, and BM25 at NDCG ranks of 1, 3, and 10. The EV-DSSM outperformed all approaches by a statistically significant margin. DSSM also achieved strong results, outperforming TF-IDF, BM25, and LSA. The performance of both the standard and extended versions indicates that the core DSSM approach is a suitable method to link exploits and vulnerabilities compared to competing approaches. Table 21 and Figure 18 summarize baseline evaluation results.

Method	NDCG@1	NDCG@3	NDCG@10
EV-DSSM	.397*	.428*	.452*
DSSM	.376	.406	.439
TF-IDF	.363	.395	.418
BM25	.306	.359	.399
LSA	.298	.305	.356

Table 21. EV-DSSM Benchmark Results

Note: * indicates statistically significant improvement over other methods ($p < 0.05$)

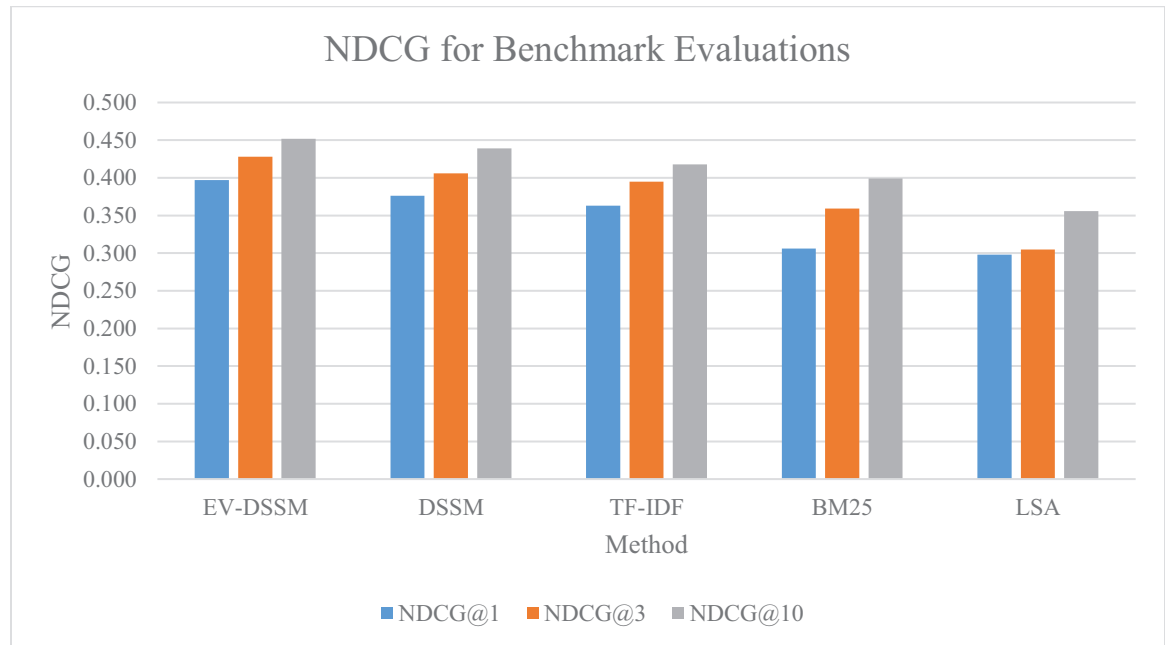


Figure 18. DSSM Benchmark Evaluation Results

Although both DSSM approaches outperformed standard techniques, the EV-DSSM outperformed the standard DSSM by a statistically significant margin. This difference indicates that EV-DSSM benefits from including contextual features (i.e., vulnerable version names) to create linkages rather than just the vulnerability and exploit names. The top of Figure 19 summarizes several examples of correct linkages found by the EV-DSSM but missed by the standard DSSM and other baselines. The bottom of Figure 19 highlights key characteristics in the vulnerability description and the hacker exploit post that helped create the appropriate link.

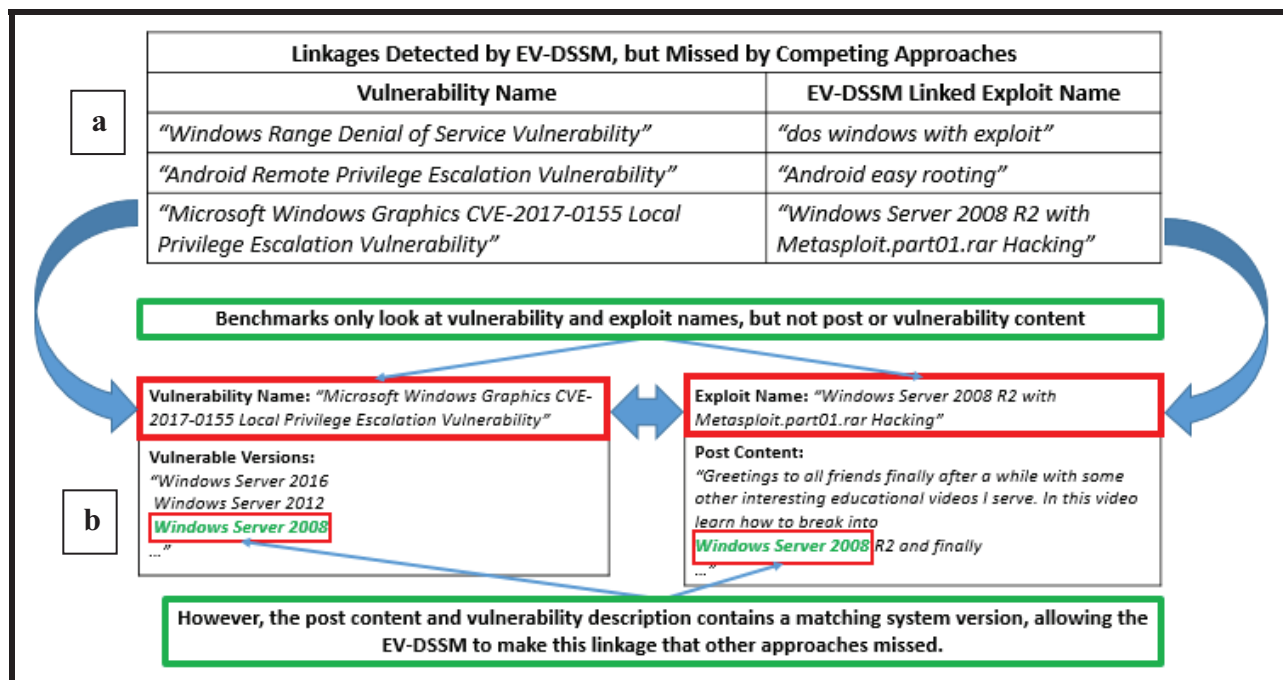


Figure 19. EV-DSSM Performance Example
(a) Example of links identified by the EV-DSSM not detected by other approaches and (b) key characteristics of the vulnerability description and exploit post that aided in creating the appropriate linkage

As illustrated in Figure 19, benchmark approaches only account the vulnerability name and exploit name when making linkages. As a result, they do not accurately recognize all linkages. The bottom of Figure 19 depicts a matching system version, "Windows Server 2008", in both the vulnerability description and the hacker exploit post. Extracting this content and appending it to the vulnerability and exploit names prior to word hashing allowed the EV-DSSM to create the correct exploit-vulnerability link that other approaches missed. Other examples listed in Figure 19 were linked due to the vulnerable version name found in both the hacker forum post and the vulnerability description.

From a domain perspective, detecting exploits specifically designed for particular vulnerabilities is crucial. CTI professionals operate in an environment in which accurate and actionable intelligence is required to make effective cybersecurity decisions. Furthermore, organizations have finite resources; not every potential threat can be investigated and mitigated

in a timely fashion. Accurately pinpointing exploits designed for specific systems can significantly improve the efficiency of investigations and mitigation strategies. The EV-DSSM offers benefits over other approaches in this regard, as it includes valuable cybersecurity contextual cues in its process.

3.5.2. EV-DSSM Word Hashing Evaluations

The second set of experiments helps further to determine the ideal word hashing method to link exploits and vulnerability names for our application. As previously mentioned, standard word hashing segments input text into letter trigrams for DSSM's in natural language applications. However, the cybersecurity context has many non-natural terms such as version numbers. Our experiments examined the performance of utilizing letter trigrams, letter bigrams, and word n-gram (i.e., unigram) for both the EV-DSSM and DSSM. Experiment results summarized in Table 22 and Figure 20 indicate that the EV-DSSM utilizing the letter trigram approach significantly outperforms all other configurations. This includes DSSM with letter trigrams, letter bigrams, and word n-grams. Within each configuration, the EV-DSSM outperformed the standard DSSM. These discoveries are consistent with the performance of the EV-DSSM in the first experiment.

Configuration	Method	NDCG@1	NDCG@3	NDCG@10
Letter Trigram	EV-DSSM	.397**	.428**	.452**
	DSSM	.376	.406	.439
Letter Bigram	EV-DSSM	.384*	.416*	.434*
	DSSM	.361	.392	.419
Word n-gram	EV-DSSM	.372*	.389*	.402*
	DSSM	.359	.373	.385

Table 22. Word Hashing Evaluation Results

Note: * indicates statistically significant improvement over other hashing methods ($p < 0.05$) ** indicates statistically significant improvement over all methods ($p < 0.05$)

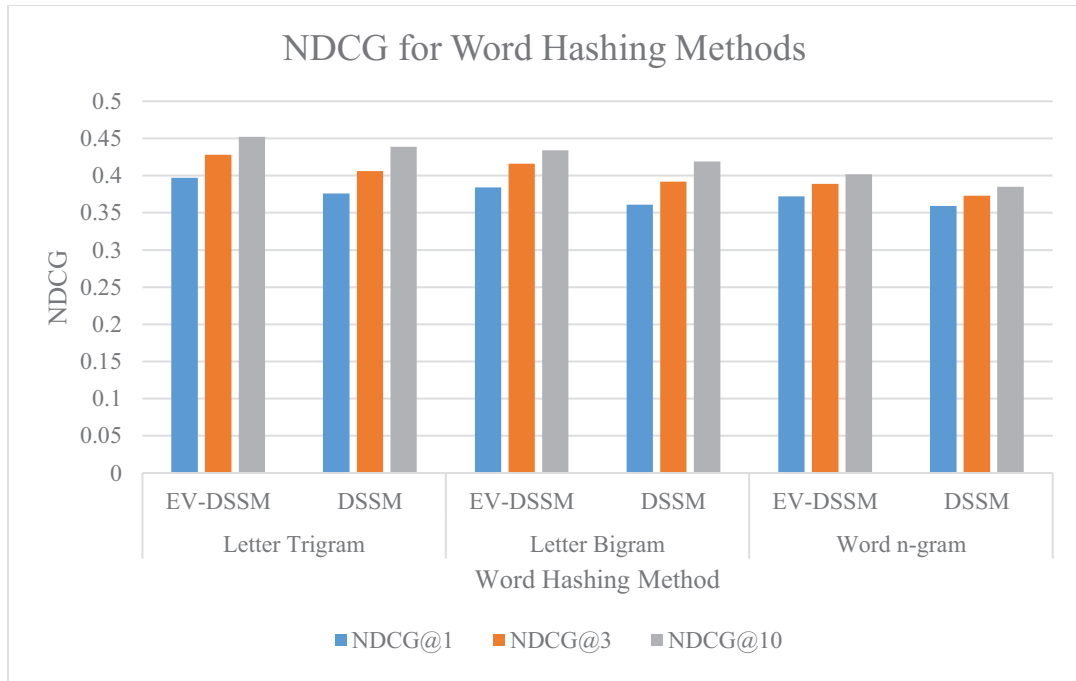


Figure 20. DSSM Word Hashing Evaluation Results

The EV-DSSM and DSSM both achieved statistically significant differences when using letter trigrams than with letter bigrams or word n-grams. These results indicate that using the letter trigram approach can extract representations that letter bigrams and word n-grams miss. For example, many exploit and vulnerability names contain three letter acronyms such as “SSH” for *Secure Shell*, “SQL” for *Structured Query Language*, “PHP” for *Hypertext Preprocessor*, “XSS” for *Cross-Site Scripting*, and many others. These acronyms are key components of vulnerability and exploit names like “*OpenSSH Security Bypass*,” “*SSH Cracker*,” “*PHP Remote Code Execution*,” and “*Casper PHP Trojan*.” Letter bigrams create too small of a window (e.g., “ph” and “hp” for “php”), while word n-grams create windows too large (e.g., “OpenSSH” instead of capturing just “SSH”). Letter trigrams create a window large enough to capture these key three letter acronyms. Future studies applying DSSM’s in non-natural contexts such as computer generated logs can use this knowledge to guide their approach and enhance their results.

3.5.3. US Hospital Case Study

Vulnerability assessment of the top eight hospitals revealed that 344/1,879 (18.31%) of scanned devices have vulnerabilities. 176 of these devices have multiple vulnerabilities, while the remaining 168 have only one. Vulnerabilities in the ‘Critical’ threshold were due to outdated PHP, OpenSSL, or Unix versions. ‘High’ and ‘Medium’ had Apache, SQL, SSH, and XSS issues. Table 23 summarizes selected vulnerabilities in the ‘Critical,’ ‘High,’ and ‘Medium’ levels. For each selected vulnerability, we list the most relevant exploit name as determined by the EV-DSSM.

Risk Level	Vulnerability Names (Severity)	Linked Exploit Name (Post Date)	Number of Devices
Critical	“PHP Unsupported Version Detection” (10.0)	“PHP Patch Bugs” (12/14/2014)	11
	“OpenSSL Unsupported” (10.0)	“SSL Poodle” (10/14/2014)	7
	“Unix OS Unsupported Version Detection” (10.0)	“Bypass Unix Virus” (10/23/2013)	6
High	“Multiple Apache Vulnerabilities” (8.3)	“Bypass Error on Apache” (10/21/2013)	17
	“SQL Injection” (8.0)	“Precision Test SQL Injection” (2/1/2016)	8
Medium	“HTTP TRACE / TRACK Methods Allowed” (5.0)	“Assassin HTTP Botnet” (5/18/2015)	58
	“SSH Weak Algorithms” (4.3)	“SSH Botnet Tool” (1/18/2015)	55
	“Multiple XSS Vulnerabilities” (4.3)	“XSS Lab” (7/27/2015)	34

Table 23. Selected Hospital Vulnerabilities Their Most Relevant Exploit Identified by the EV-DSSM

After the EV-DSSM determined the most relevant exploit for each vulnerability, we calculated severity scores for devices with multiple vulnerabilities. Doing so allows CTI professionals to prioritize devices on their networks that need immediate remediation. While it is impossible to validate whether the EV-DSSM exploit can take advantage of the detected vulnerability without actually executing the exploit, the provided information can offer a

promising starting point for mitigation activities. For space considerations, we only list the top ranked device on each hospital's network in Table 24. We also anonymize the last three octets of each hospital's IP range and selected device to protect their privacy.


Selected Devices for Each Hospital			Severity Score Information		
Hospital IP Range	IP Address	Device Type	# of Vulnerabilities	Vulnerabilities	DVSM
12x.x.x.x	12x.x.x.x	FTP/SSH Server	3	FTP issues	3.233
19x.x.x.x	19x.x.x.x	SSH Server	3	SSH issues	3.193
17x.x.x.x	17x.x.x.x	eCare web portal	47	XSS, OpenSSL, buffer overflow, DoS	56.398
16x.x.x.x	16x.x.x.x	Medical computing portal	5	PHP and SSH issues	4.863
14x.x.x.x and 14x.x.x.x	14x.x.x.x	Web Server	3	SQL Injections	8.009
	14x.x.x.x	Apple TV	2	Buffer overflow	6.909
14x.x.x.x	14x.x.x.x	SSH/Web server	4	PHP and SSH issues	4.088
6x.x.x.x	6x.x.x.x	Informational diabetes portal	3	Unix vulnerabilities	6.616
16x.x.x.x	16x.x.x.x	Web Server	6	XSS	8.233

Table 24. Most Susceptible Device on Each Hospital's Network

Table 24 indicates that SSH servers, web servers, an Apple TV, and medical portals that are all susceptible to exploitation. All but one device had between two and six vulnerabilities due to various web application, SSH, and outdated software issues. Hackers can potentially exploit any of these vulnerabilities with the selected tools to gain a foothold into the hospital's network. From there, they can follow well-documented procedures of moving laterally within the network to discover other exploitable devices (Weidman 2014).

The most susceptible device from our analysis was an eCare portal on the 17x.x.x.x network (DVSM 56.398). While we cannot confirm the portal's exact purpose due to its login

requirements, it can be surmised that the portal provides various healthcare related services to patients and/or employees of the hospital. Figure 21 depicts the system's web interface, selected vulnerabilities, most relevant exploit name for each vulnerability, the individual severity score for each exploit-vulnerability link, and the overall device vulnerability score.



Vulnerability Name (CVSS Score)	Exploit Name (Post Date)	Severity Score
"OpenSSL Unsupported" (10.0)	"SSL Poodle" (10/14/2014)	3.352
"Multiple XSS Vulnerabilities" (4.3)	"XSS Lab" (7/27/2015)	1.519
...
-	-	Total: 56.398

Figure 21. Selected Vulnerabilities from Partners eCare Portal on the 17.x.x.x Hospital Network

Nessus detected 47 vulnerabilities for this device resulting in an overall DVSM score of 56.398. The large and diverse nature of this device's attack surface increases its exploit probability. Vulnerabilities in this device include cross-site scripting, OpenSSL issues, buffer overflow, and denial of service. This device also offers a login form, indicating that it connects to a database. Hackers can exploit the form to access the underlying database and gain a valuable foothold into the hospital's network to pivot to other devices. All of these weak points can allow an attacker to remotely take the system offline or hijack it for their own use (Weidman 2014).

3.5.4. SCADA Device Case Study

As mentioned in our research framework, the SCADA device case study aims to illustrate how CTI professionals can apply our framework to not only prioritize devices on a network, but also identify systemic vulnerabilities and their relevant hacker exploits for a category of devices across society. The Nessus vulnerability assessment results for the SCADA case study showed that 4,009/20,461 (19.59%) of the tested devices have 'Critical' (182), 'High' (189), 'Medium' (2,737), or 'Low' (901) risks. The majority of the vulnerabilities are

due to Unencrypted Telnet Servers and SSH Server vulnerabilities. Table 25 summarizes these vulnerabilities, their severities, most relevant exploit name, and number of affected devices and major afflicted vendors of devices containing these vulnerabilities.

Vulnerability Name (Severity)	Exploit Name (Post Date)	# of Afflicted Devices	Afflicted Vendors
“Unencrypted Telnet Server” (5.8)	“Cracking Telnet with Brutus” (8/8/2012)	1,407	Rockwell Automation/ABB, Siemens, Schneider Electric, Power Measurement, Acromag, Honeywell
“Dropbear SSH Server Vulnerabilities” (5.0)	“SSH Botnet Tool” (1/18/2015)	524	

Table 25. Selected SCADA Vulnerabilities Their Most Relevant Exploit as Identified by the EV-DSSM

Analysis reveals that the vulnerabilities listed in Table 25 affect SCADA devices such as Programmable Logic Controllers (PLC’s) from major vendors including Rockwell Automation, Siemens, and Schneider Electric. PLC’s are computers that automate and monitor electromechanical processes such as electrical relays, hydraulics, and motors. These components often appear in factories and industrial heating and cooling units. Exploiting the Telnet or SSH vulnerability on these devices would allow hackers to remotely control the device. From there, hackers can adjust critical operations to damage or incapacitate the PLC and any device it controls.

Specifically examining the Telnet vulnerability-exploit information indicates that hackers can exploit the Telnet protocol to not only gain access over the system, but to also monitor and alter communications to and from the system made by other individuals. Such communications may allow them to better understand how the PLC is operated and help attackers to conceal their attacks. In the case of the SSH vulnerability, the SSH botnet tool identified by the EV-DSSM indicates that a skilled hacker can potentially exploit many PLC’s simultaneously and turn them into bots to attack other targets. This type of attack is not

unprecedented; the Mirai malware infected hundreds of thousands of vulnerable IoT devices to conduct a large-scale Distributed Denial of Service (DDoS) against the Internet's DNS servers in 2016 (Mathews 2016). Given the widespread nature of this vulnerability across the tested SCADA devices, vendors and device owners alike should take steps to investigate whether their systems are truly susceptible to identified vulnerabilities and proceed with the appropriate mitigation activities.

3.6. Conclusion and Future Directions

Preventing cyber-attacks is one of modern society's grand challenges. CTI offers organizations the opportunity to mitigate cyber-attacks before they occur. However, many organizations struggle to implement effective CTI capabilities due to their inability to pinpoint relevant exploits for their vulnerabilities. Hacker forums provide a novel data source that, when coupled with known vulnerabilities, can develop proactive and holistic CTI. Although IS scholars are uniquely equipped with behavioral, economic, and design science paradigms to produce significant CTI research contributions, existing cybersecurity literature focuses primarily on behavioral compliance, risk management, investments in securing digital assets, and market effects of securing digital assets (Hui et al. 2016). These current focus areas, combined with existing CTI limitations, has resulted in IS scholars calling for an increased emphasis on cybersecurity analytics and black hat hacker studies (Chen et al. 2012; Mahmood et al. 2010).

In this essay, we adopted design science principles to create a novel computational IT artifact known as the EV-DSSM method to automatically link hacker exploits to vulnerabilities. The EV-DSSM's design novelty of including key exploit and vulnerability cues was guided by the key CTI principles of relevance and timeliness. Rigorous evaluation of the

EV-DSSM demonstrates its statistically significant performances over the standard DSSM and other baselines. Although the EV-DSSM's (and DVSM's) proof-of-concept and proof-of-value was illustrated using openly accessible US hospital devices and SCADA systems, numerous scholars have posited that novel IT artifacts should contribute knowledge to the IS knowledge base to guide future IS research (Gregor and Hevner 2013; Hevner et al. 2004; Nunamaker et al. 1990; Peffers et al. 2007). Other than the instantiation of the EV-DSSM method for CTI, our theoretical contribution is the provision of three design principles: (1) richer semantic representations to enable accurate and relevant linkages, (2) capturing the decay in value of an exploit via mathematical representations, and (3) ideal word hashing representation of cybersecurity text. Each provides novel prescriptive knowledge to the IS knowledge base can be applied to other cybersecurity data analytics contexts in the form of nascent design theory (Gregor and Hevner 2013).

We believe this study is an important first step for IS scholars to contribute holistic CTI knowledge and approaches by studying black hat hacker exploits and employing cybersecurity analytics. As with any study examining a phenomenon for the first time, there are limitations with our approach. For example, we cannot validate whether the detected vulnerabilities are truly susceptible to the linked hacker exploits. Actual validation would require us to exploit the detected vulnerabilities, an act with significant legal and ethical ramifications. Additionally, we are limited to the types of vulnerabilities we can link hacker exploits to those only listed by Security Focus. In spite of these limitations, the EV-DSSM and DVSM and the design principles used to create them open several promising directions research avenues for IS scholars. While past research has discovered that hackers from various geo-political regions (e.g., Russia, Middle East, etc.) have different motivations, the types of systems they target

and the exploits they use is unclear. Researchers can apply the EV-DSSM and DVSM across multiple geo-political regions to pinpoint these differences. Behavioral IS scholars can use general theories of cyber-crime to help understand why hackers target specific types of vulnerabilities. Each direction would help improve the depth and breadth of cybersecurity studies conducted by IS scholars and ultimately, provide a safer and more secure society.

4. ESSAY III: IDENTIFYING EXPLOIT SHARING HACKERS AND COMMUNITIES: A GRAPH CONVOLUTIONAL AUTOENCODER APPROACH

4.1. Introduction

The pervasive use of information technology has provided modern society with remarkable benefits. Many organizations across industry, government, and academia employ databases, communication networks, and other systems to produce societally impactful services and products. Unfortunately, many of these technologies are also prime targets for malicious hackers across the globe. Recent estimates place the total cost of cyberwarfare, hacktivism, espionage, and other hacking activities against major organizations such as Equifax, Uber, and Yahoo! at \$450 billion annually (Graham 2017). Today, many organizations and law enforcement agencies aim to identify and understand malicious hackers conducting cyber-attacks through the development and use of Cyber Threat Intelligence (CTI).

CTI is fundamentally a data-driven process that develops relevant, timely, and actionable intelligence about emerging threats and key threat actors (i.e., hackers) to enable effective cybersecurity decisions. Traditionally, CTI has focused on collecting data generated from Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and log files from databases, servers, and firewalls. Well-refined analytics such as malware analysis, anomaly detection, event correlation, forensics, and others helps profile hackers based on their attack preferences (e.g., ransomware, SQL injections), and skill-level (e.g., script kiddy). Despite the maturity of these procedures, CTI experts from the world-renowned SANS Institute note that “most organizations are still reactive to alerts and incidents instead of

proactively seeking out the threats” (Lee and Lee 2017). Consequently, the volume, severity, and sophistication of cyber-attacks are on a steady and unfortunate increase.

To alleviate these concerns, CTI experts suggest proactively studying hackers in the vast and evolving international online hacker community (Bromiley 2016; Shackleford 2016). The online hacker community is an attractive CTI data source as it motivates millions of hackers from the US, Russia, Middle East, and China to share exploits and knowledge on four major platforms: DarkNet Markets, carding shops, hacker forums, and Internet-Relay-Chat (IRC). Among these, hacker forums are of particular interest amongst CTI experts as it provides unique mechanisms that allow hackers to freely share and discuss malicious exploits. Figure 22 illustrates one such example, where a one hacker requests a SQL injection tool and another hacker provides it.

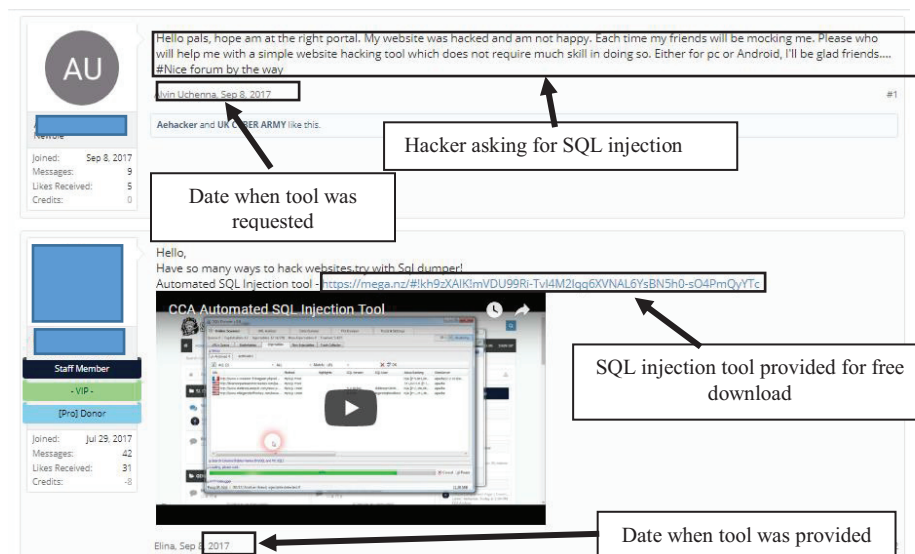


Figure 22. Example of Exploit Sharing in Hacker Forum
Example of a hacker (top) asking for an SQL Injection and receiving it from another (bottom)

Hackers have used forums to develop exploits used in well-known breaches. One example is Target, where hackers discussed, developed, and refined the BlackPOS malware in forums months before the attack (Kitten 2014). Such events motivate the careful examination

of forums to identify key hackers and exploit sharing communities. However, analyzing hacker forums is a non-trivial task. Hacker forums contain hundreds of thousands of unstructured, unsanitized text records. Traditional CTI analytics to identify and group hackers were not designed to handle these data characteristics. Moreover, popular approaches of identifying key actors and communities in social media contexts, such as Social Network Analysis (SNA) require significant adjustments to generate valuable CTI. These challenges motivate the development of novel CTI analytics.

In this study, we develop novel proactive CTI capabilities by adopting an SNA approach to explore online hacker forums for two purposes: identify exploit sharing communities and key hackers. To achieve the first goal, we develop a novel deep learning architecture, the Graph Convolutional Autoencoder (GCAE). The GCAE integrates the core operations of an emerging method, the Graph Convolutional Network (GCN), into the popular autoencoder architecture to develop low-dimensional, generalized embeddings for each node within a hacker’s social network based on its structural (i.e., who each hacker is connected to) and nodal (i.e., exploit post content) attributes in an unsupervised fashion. Through a series of benchmark experiments, we demonstrate that embeddings generated from the GCAE outperforms state-of-the-art node embedding approaches in detecting exploit sharing communities within hacker forums. To execute the second goal, we extend the classic degree centrality measure commonly used in SNA literature to create the Exploit Degree Centrality (EDC). The EDC incorporates valuable hacker exploit temporal features to accurately pinpoint current key hackers within forums. Both the GCAE’s and EDC’s utility are demonstrated with an in-depth case study of a large English hacker forum.

The remainder of this essay is organized as follows. First, we review literature related to the online hacker community, SNA, GCN's, and autoencoders. Second, we identify existing gaps within reviewed literature and posit questions for study. Third, we detail our research design and testbed. Subsequently, we summarize experimental procedure and results. The practical value of our approach is then demonstrated via an in-depth case study of a large English hacker forum. We conclude this work by summarizing our contributions and highlighting promising directions for future work.

4.2. Literature Review

Four areas of literature are searched and examined: (1) hacker community research to discover key forum features and how past work has detected key hackers and/or communities, (2) SNA to review approaches in identifying key actors and communities within networks, (3) GCN's to understand its core operations and applications, and (4) autoencoders to guide GCAE's development.

4.2.1. Hacker Community Research

As noted in the introduction, hackers across major geo-political regions (e.g., Russia, US, China, Middle East, etc.) congregate on four online platforms to share malicious knowledge and tools: IRC channels, carding shops, DarkNet Marketplaces, and forums (Benjamin et al. 2015; Goel 2011; Samtani et al. 2017). While each platform offers significant CTI value, hacker forums are particularly useful for pinpointing hackers sharing malicious exploits (Hutchings and Holt 2015; Li et al. 2016; Samtani et al. 2017; Samtani et al. 2016; Sood and Enbody 2013; Zhao et al. 2016). IRC channels and carding shops do not have the mechanisms for hackers to post and access exploits, while DarkNet Markets are not entirely focused on cybersecurity content. Moreover, forums provide post dates and author information

for each posted exploit, valuable metadata for generating CTI that are not consistently available in other platforms. These characteristics have led to substantial research inquiries focused on identifying key hackers and/or communities within forums. Table 26 summarizes selected studies.

Year	Author	Data Source	Focus	Key Hacker ID Method	Metric to Identify Key Hackers	Community Detection?
2017	Samtani et al.	Eight forums	ID exploit categories and key hackers for crypters	SNA	Standard Centralities	No
2017	Macdonald and Frank	One forum	ID key hackers and communities sharing malware	SNA	Standard centralities	Yes
2017	Grisham et al.	Two Arabic, one Russian, one English forum	ID key mobile malware sharing hackers	SNA	Standard centralities	No
2016	Huang and Chen	Baidu Tieba	ID key hackers	SNA	Standard centralities	No
2016	Samtani and Chen	One English forum	ID key hackers sharing keyloggers	SNA	Standard centralities	No
2014	Li and Chen	One Russian forum	ID key carding sellers	Sentiment analysis	Sentiment ratings	No
2012	Holt et al.	Samples from 8 Russian forums	ID roles of hackers in forums	SNA	Standard centralities	No

Table 26. Selected Hacker Forum Literature Identifying Key Hackers and/or Communities

SNA's ability to capture relationships between hackers, combined with its suite of centrality measures (detailed in following subsection), has made it the preferred methodology to identify key hackers. To date, scholars have employed SNA to identify key hackers for general malware (Macdonald and Frank 2017), mobile malware (Grisham et al. 2017), crypters (Samtani et al. 2017), keyloggers (Samtani and Chen 2016), or hackers in general (Huang and Chen 2016; Holt et al. 2012). Despite useful applications, existing literature has two key limitations preventing valuable CTI development. First, the centrality measures (specifically degree) used to pinpoint key hackers are often misleading in CTI applications. For example, a hacker sharing four exploits in 2011 would have the same ranking as one sharing four exploits in 2017. While this may suffice in other applications, CTI is a domain in which timeliness is essential; exploits exponentially decay in value as time elapses (FIRST 2017). Thus, a hacker posting four exploits in 2017 compared to four in 2011 should receive more weightage due to the recency of their contributions.

The second major limitation is the lack of work identifying hacker communities within forums. Oftentimes, valuable and actionable CTI is not simply pointing out one hacker, but a discovering group of hackers with similar skill-sets, expertise, and/or tool specialties (Friedman 2015). Such knowledge can offer valuable insights on hacker Tools, Techniques, and Procedures (TTP) to help develop holistic cyber-defenses. Moreover, it can pinpoint hackers who have the capability to execute a particular attack type (e.g., ransomware). In light of these limitations, SNA fundamentals are reviewed for two purposes: (1) understand how standard SNA centrality measures can be extended to encompass temporal cues from hacker forums and (2) identify community detection methods that can pinpoint exploit sharing communities.

4.2.2. Social Network Analysis (SNA): Centrality Measures and Community Detection

SNA builds upon graph theory to study relationships between social entities. Social networks are constructed with two building blocks: nodes and edges. Nodes represent social entities (e.g., people, organizations, etc.) and edges denote relationships between nodes. Various node and edge configurations has allowed researchers to study information diffusion, network evolution, and other phenomena across multiple social media contexts (Barabasi 2016). SNA's flexibility is due in large part to its robust mathematical representation for node relationships. In its standard set up, a network G containing N nodes has an adjacency matrix A with N rows and N columns. If a relationship exists between nodes i and j , the value of element A_{ij} is denoted as 1. If no relationship exists, element A_{ij} remains 0. This representation allows researchers to calculate numerous descriptive statistics (e.g., density, length, diameter, radius, etc.) to understand the overall network. More importantly, they enable the calculation of four centrality measures (summarized in Table 27), each of which offers value in identifying key nodes within a network.

Centrality Measure	Description	Measure Type	Formula	Value
Degree	Number of links leading in and out of a node. Can be decomposed to in and out-degree for directed networks	Count based	$c_i = \sum_j a_{ij}$	Measures immediate influence
Eigenvector	Summed connections to other nodes weighted by their centralities		$x_i = \frac{1}{\lambda} \sum_j a_{ij} x_j$	Measures how well connected a node is
Closeness	Average number of hops required to every other node on the network (i.e., sum of all distances to other nodes)	Location based	$c_i = \sum_j d_{ij}$	Measures how quickly a node can reach others
Betweenness	Number of shortest paths passing through a node divided by all shortest paths		$b_k = \sum_{i,j} \frac{g_{ikj}}{g_{ij}}$	Measure importance of social position

Table 27. Standard Centrality Measures Used to Identify Key Nodes in a Network

The four centrality measures described in Table 27 are either count (degree and eigenvector) or distance (closeness and betweenness) based. Count based methods tally the number of connections (weighted or unweighted) leading to and from a node. Distance based centralities measure how close nodes are from each other (e.g., number of hops). Among the four measures, degree centrality is the most widely used when pinpointing key nodes in a network. Degree centrality's importance has led researchers to construct diagonal matrix D , where the diagonal represents the degree of each node within the network. Both the degree and adjacency matrix described above Table 27 are key ingredients for another common SNA activity: community detection.

Community detection aims to identify groups of nodes sharing similar properties (Barabasi 2016). Examples of communities within networks include similar websites on the World Wide Web (WWW), clubs within a university network, and proteins by function within a protein network. Ideally, nodes in each group are more related to each other (in terms of relationships, interests, specialties, etc.) than to nodes in other groups (Barabasi 2016). Two major approaches of community detection algorithms exist: graph partitioning and node similarity. Graph partitioning algorithms, such as Girvan-Newman, Karypis-Kumar, and clique percolation, iteratively remove edges until the number of edges within each community exceeds the outgoing edge count. Node similarity methods use clustering algorithms such as k-means, spectral, or agglomerative to group nodes based on their structural features (embeddings).

Between the two, node similarity approaches are commonly used in social media networks (Barabasi 2016). Nodes in such networks often have attributes (e.g., join date, number of posts, interests, etc.). Scholars have empirically proven that incorporating node

attributes improves community detection accuracy (Yang et al. 2014). However, these studies only account for a limited number of categorical variables; not high-dimensional feature vectors (Jia et al. 2017). In the context of hacker forums, scholars have noted that including that supplementing each node’s structural properties with their textual exploit posts (high-dimensional in nature) can create finer-grained, more accurate communities of hackers with similar specialties (Macdonald and Frank 2017; Samtani et al. 2017). This, combined with the current limitation in node-attributed community detection necessitates a robust method that can combine both structural and high-dimensional textual nodal attributes. One emerging method that can offer value in such a task is the deep learning based GCN, which is reviewed next.

4.2.3. Graph Convolutional Networks (GCN)

GCN’s leverage recent advances in deep learning to learn embeddings from node-attributed networks for supervised learning tasks (Kipf and Welling 2017). The GCN learns a function $f(X, A)$, where X is an $N \times F$ input node feature matrix (i.e., feature vector for each node, can be text attributes) and A is the $N \times N$ adjacency matrix from graph G . The GCN outputs Z , an $N \times F$ feature matrix that encompasses both the nodal and structural features for each node in G , where F is the dimension of output feature vectors. While A and X could be concatenated and inputted into an Artificial Neural Network (ANN), doing so results in a huge number of parameters and requires ANN retraining if the number of nodes change. Recognizing these issues, Kipf and Welling (2017) formulated $f(X, A)$ to employ a more complicated layer-wise propagation function (based on the ANN):

$$H^{l+1} = \sigma(AH^{(l)}W^{(l)})$$

Where σ is an activation function (usually Rectified Linear Unit (ReLU)), A is the adjacency matrix for G , $H^{(l)}$ is the row-wise embedding of the graph nodes in the l th layer ($H^{(0)}$ is X), and $W^{(l)}$ is the a weight parameter matrix. Although this formulation accounts for both the structural and individual features of a node, multiplying by A has two issues. First, each node's feature vectors for all its neighboring nodes are summed. However, this summation does not include the node itself, resulting in a loss of information. Second, multiplying by A changes the scale of the outputted feature vectors. To overcome the former, self-loops are captured by adding the identity matrix I_N to A to form \tilde{A} . For the scaling issue, a symmetric normalization is applied by using $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, where D is the node degree matrix. These two adjustments result in a generalized GCN propagation rule:

$$H^{l+1} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

Where \hat{A} is $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. In addition to addressing the two above-mentioned issues, Kipf and Welling (2017) mathematically proved that this formulation is a first-order approximation of spectral graph convolution, where it convolves around each node's neighborhood to capture a rich set of information. Consequently, it simulates the convolution and pooling operations of a traditional Convolutional Neural Network (CNN). Like the standard CNN, the GCN is trained with a gold-standard dataset. The error between the predicted output and the ground truth backpropagates through the GCN, recalibrating the weights accordingly. The robustness of GCN operations have enabled scholars to pursue numerous research inquiries, summarized in Table 28.

Year	Author	Dataset(s)	Task	Network Nodes	Node Attributes
2018	Chen et al.	Cora, Pubmed, Reddit	Node classification	Documents	Term frequencies
2017	Manessi et al.	DBLP	Node classification	Authors	# of articles published
2017	Kipf et al.	Cora, Citeseer, Pubmed	Link prediction	Documents	Term frequencies
2017	Hamilton et al.	Citation, Reddit, PPI	Node classification	Documents	Term frequencies
2017	Yasunaga et al.	Data Understanding Conferences Papers	Document classification	Sentences	Sentence embedding
2017	Demeril	Brown Corpus	Part-of-Speech classification	Words	Word embedding
2017	Kipf and Welling	Citeseer, Cora, Pubmed, Nell	Semi-supervised classification	Documents	Term frequencies

Table 28. Selected Studies Using Graph Convolutional Networks

Scholars have leveraged GCN's ability of incorporating network structure and nodal attributes to achieve state-of-the-art performances in link prediction (Kipf et al. 2017), node classification (Chen et al. 2018; Manessi et al. 2017; Hamilton et al. 2017), document classification (Yasunaga et al. 2017) part-of-speech classification (Demeril 2017), and semi-supervised classification (Kipf and Welling 2017). Despite GCN's breadth of applications, it remains a supervised learning approach. Thus, the embeddings learned from the network the GCN process are task-specific; they are the most salient features that create the best mappings between the input data and pre-specified output labels. As a result, these node embeddings are less suitable for and generalizable to other tasks (e.g., community detection). Given the focus of this study, this drawback necessitates the adaptation of the core GCN operations into a deep learning architecture that can create generalized node embeddings without a gold-standard dataset. One such architecture is the autoencoder, which is reviewed next.

4.2.4. Autoencoders

Commonly used in dimensionality reduction, the autoencoder is an unsupervised deep learning algorithm that learns a low-dimensional embedding for a data input (Goodfellow et al. 2016). The autoencoder's structure follows that of a standard, feed-forward ANN. Unlike the ANN, the autoencoder aims to reproduce its input at the output layer. The autoencoder has three components, depicted in Figure 23: the encoder, the compressed feature vector (i.e., embedding), and decoder.

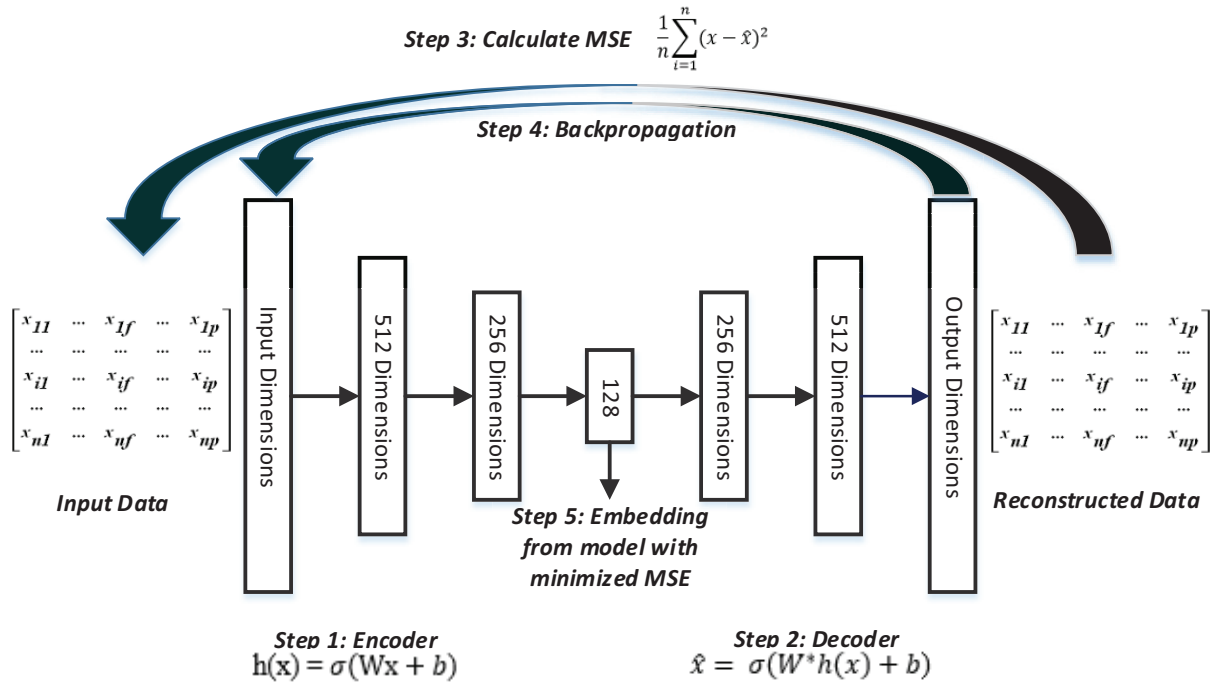


Figure 23. Autoencoder Architecture

From left: Step 1 reduces dimensionality of inputted data until a bottleneck is reached.

Step 2 aims to reconstruct output from the bottlenecked representation. Step 3 compares reconstructed output with original input. Step 4 uses backpropagation to adjust autoencoder weights. The process continues until error between output and input is minimized.

The encoder learns a function $h(x) = \sigma(Wx + b)$ where σ is the activation function, W is the weight matrix for layer l in the encoder, x is the data input, and b is the bias term. Each encoder layer steadily reduces the dimensionality of the data until a bottleneck is reached

(center of Figure 23). The decoder then expands the dimensionality of the embedding generated by the encoder and attempts to recreate the input with a function $\hat{x} = \sigma(W^*h(x) + b)$ where W^* is the weight matrix for layer l in the decoder, $h(x)$ is the embedding generated from the encoder (i.e., $\sigma(Wx + b)$), and b is the bias term. The error calculated between the reconstructed output \hat{x} and original input x is backpropagated through the autoencoder to adjust weights. This weight adjustment procedure mimics the ANN process, where the ground-truth labels are compared to the predicted output and the ANN weights are tuned. The overall process of encoding, decoding, error calculation, and weight adjustment continues until the error between x and \hat{x} is minimized. The compressed feature vector in the bottleneck of the autoencoder with minimal error is a generalized embedding that can support numerous clustering and classification tasks.

4.3. Research Gaps and Questions

Our literature review revealed several gaps. First, standard SNA centralities used to identify key hackers do not account for the age of the exploits they share. As a result, they may inaccurately identify key exploit sharing hackers, preventing the development of valuable CTI. Second, minimal hacker forum literature has aimed to identify communities of exploit sharing hackers in forums, a task that can help create comprehensive cyber-defenses. However, accounting for both a network's structure and nodal attributes to create generalized embeddings for community detection is a non-trivial task. While GCN's have achieved state-of-the-art performances, their current structure generates task-specific node embeddings. Autoencoders offer much promise in creating task-independent, general node embeddings in an unsupervised manner, but it is unclear how their core structure can be modified to handle

graph convolution operations. These domain and methodological knowledge gaps motivate the following research questions:

- How can the standard centrality incorporate temporal features to identify key exploit sharing hackers?
- What communities of exploit sharing hackers exist within online hacker forums?
- How can the autoencoder be adapted to incorporate graph convolution operations to generate task-independent embeddings for community detection applications?

4.4. Research Design and Testbed

To address the identified research gaps and answer the posed research questions, a novel CTI research framework (Figure 24) with four major components was developed: (1) Data Collection and Pre-Processing, (2) Network Representation, (3) GCAE Construction, and (4) Key Hacker and Community Detection. Each component is summarized in the following subsections.

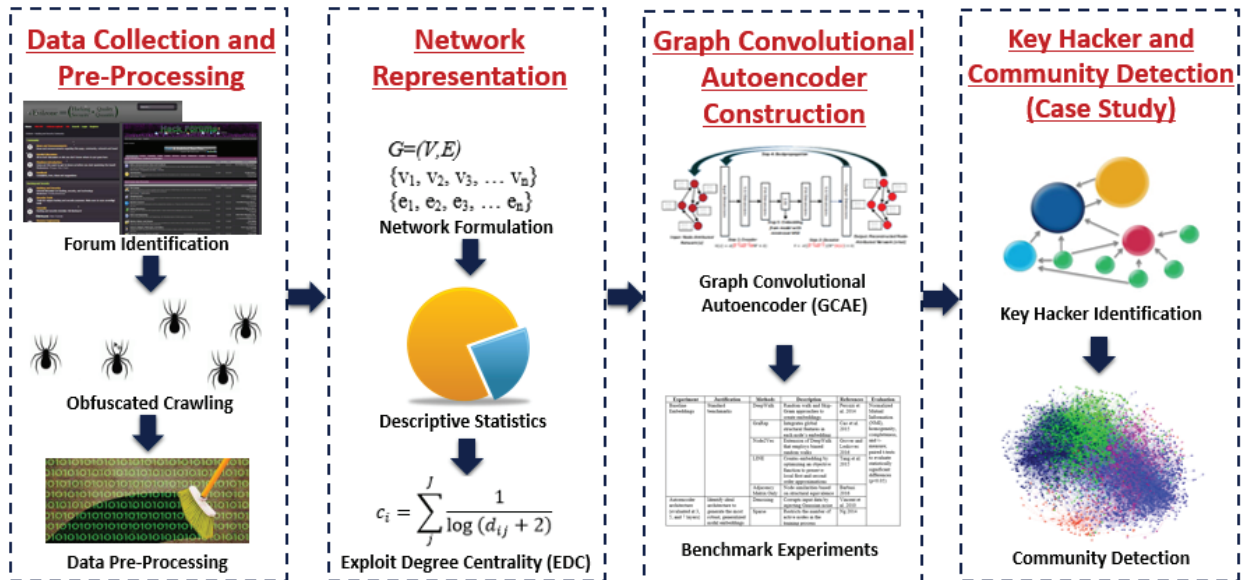


Figure 24. Research Framework for Identifying Key Exploit Sharing Hackers and Communities

4.4.1. Data Collection and Pre-Processing

The data collection component of our framework aims to collect two large and long-standing English hacker forums with numerous exploits. These forums were selected for several reasons. First, several cybersecurity experts suggested them for examination. Second, these forums are notorious within the online hacker community for containing a plethora of malicious exploits made by thousands of hackers. Third, these forums can be accessed without an invitation, thus allowing us to get platform access without direct hacker interaction.

Following forum identification, a web crawler routed through the Tor network downloaded all forum HTML pages onto our local disks. A customized parser using Regular Expressions sifts through the HTML pages and parses all post content, post dates, exploits, threads, and author information into a relational database. Table 29 summarizes the final collection. Forum names are anonymized to ensure we protect ourselves against malicious attacks from hackers in these communities.

Forum	Language	Date Range	# of Posts	# of Members	# of Exploits
O****C	English	2/07/2005 – 10/22/2017	137,664	7,011	2,720
T****u	English	6/10/2006 – 12/15/2016	40,666	2,539	2,206
Total:	-	2/07/2005- 10/22/2017	178,330	9,550	4,926

Table 29. Summary of Hacker Forum Collection

Our collection procedures resulted in a dataset containing 178,330 posts (4,926 exploits) made by 9,550 hackers between 2005 and 2017. Collected exploits include botnets, Denial of Service (DoS), Distributed Denial of Service (DDoS), Zeus malware, buffer overflow, keyloggers, and crypters. All posts are pre-processed by removing special characters and stop-words.

4.4.2. Network Representation: Formulation, Descriptive Statistics, and Exploit Degree Centrality (EDC)

CTI professionals often aim to gain a complete understanding of all hackers creating, sharing, and using exploits. Such knowledge enables the development of robust and comprehensive cyber-defenses against an array of potential attack vectors and hackers (Caltagirone et al. 2013). Guided by these principles, we formally denote the network as $G=(V,E)$. G is a directed graph, V is the node set, $\{v_1, v_2, v_3, \dots v_n\}$, of all hackers in a thread with an exploit, and E represents the edge set, $\{e_1, e_2, e_3, \dots e_n\}$ of directed edges from a hacker sharing an exploit to all other hackers in a thread receiving the exploit. While we recognize other representations exist, this formulation captures all hackers within the exploit sharing community without including hackers engaging in general, unrelated forum conversations.

In addition to understanding overall network dynamics with descriptive statistics such as network diameter, density, eccentricity, and average path, studies employing SNA calculate centrality measures to identify key nodes within the network. However, as noted in the literature review, standard centrality measures require inclusion of temporal features to provide valuable CTI insights. In the current network formulation, degree centrality is most conducive for extension as it measures the number of exploits contributed (out-degree) and consumed (in-degree) by a hacker. Distance-based metrics (closeness and betweenness) do not capture this information. Rather, they focus on identifying the importance of a node based on its location within the network, rather than the amount of content (i.e., exploits) it provides or receives. Given these factors, combined with degree centrality's widespread usage, we extend the standard degree centrality measure to the Exploit Degree Centrality (EDC). EDC is formally denoted as:

$$c_i = \sum_j^J \frac{1}{\log(d_{ij} + 2)}$$

Where d_{ij} is the number of days since the exploit shared between two hackers, i and j , was posted (end date 11/1/2017), creating a decaying effect of the inverse log function. c_i is the total number of in and out-links for node i in the network. Although other functions can be applied to the age, the inverse log best captures the exponential loss of value exploits have after they have been posted (FIRST 2017). Like degree centrality, EDC can be decomposed to in-EDC and out-EDC. Hackers with a higher in-EDC than out-EDC are more likely to be recent exploit consumers. If out-EDC exceeds in-EDC, then hackers are recent exploit contributors. Irrespective of breakdown, the EDC is higher if the exploits shared or consumed by a hacker are newer. Generally speaking, contributors are likely the individuals who developed the exploit and are sharing it with others to use. Consumers are more likely to be hackers at a lower skill level, but are interested in acquiring exploits to execute cyber-attacks. Identifying hackers at this level of granularity enables CTI professionals and law enforcement to execute subsequent investigations in a targeted, informed, and thus resource efficient manner.

4.4.3. Graph Convolutional Autoencoder (GCAE)

Identifying hackers with similar exploit interests (i.e., communities) can shed light on groups of individuals to monitor for both CTI and law enforcement purposes. As noted in the SNA literature review, community detection requires calculating the similarity of all nodes in a network. While this similarity has traditionally relied on the structural equivalence of nodes, both hacker forum and SNA literature indicate that incorporating nodal attributes (i.e., exploit post content) results in a comprehensive representation of each node and in turn, more accurate community detection. However, incorporating a node's structural and textual features into a

single representation is a non-trivial task. Simple concatenation of both feature sets results in a high-dimensional vector containing many irrelevant attributes that drops overall performance. While GCN's offer value a sophisticated approach of combining both feature sets, their current formulation learns the most salient features to categorize nodes into a pre-defined label. Thus, the learned representations are task-specific, and not a general representation for effective community detection. These drawbacks, combined with the autoencoder's proven ability to generate task-independent embeddings, motivates us to develop a novel deep learning architecture: the GCAE. The GCAE creates a general (i.e., task independent) embedding for each node structural and nodal attributes by extending the autoencoder's core procedures to incorporate graph convolution operations. Figure 25 depicts GCAE's architecture.

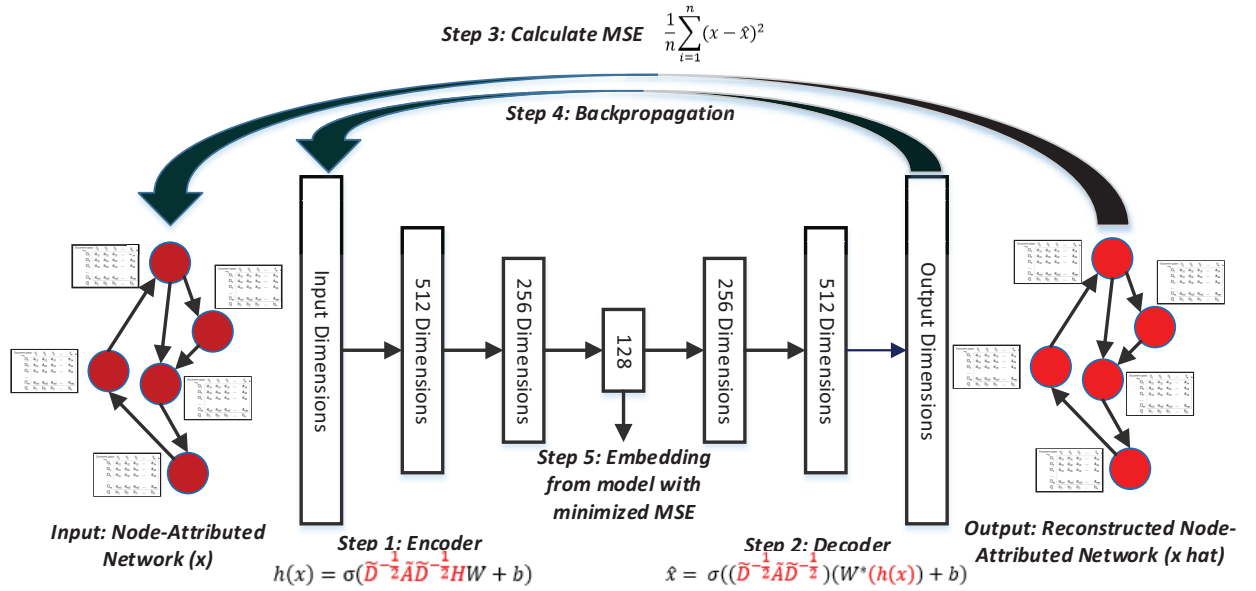


Figure 25. Architecture of the Graph Convolutional Autoencoder (GCAE)
GCAE receives adjacency and feature matrices as input and encodes them into a condensed representation using graph convolutional procedures. The decoder uses the embedding to reconstruct the input through graph convolutions. The reconstructed network and the error is backpropagated through the GCAE to update weights. The process continues until MSE is minimized.

The GCAE operates as follows. Like the GCN, GCAE’s input is A , the $N \times N$ adjacency matrix from the node-attributed graph G , and X , the $N \times F$ node feature matrix (i.e., feature vector for each node), where F is the dimension of input node feature vectors. Given our goal of identifying specific exploit sharing communities, each node’s (i.e., hacker’s) posts are represented with a term frequency vector. Previous literature has shown that employing term frequencies when representing or categorizing a set of posts with technical content such as malicious hacker exploits (Benjamin and Chen 2014; Samtani et al. 2017) and source code (Chen et al. 2015) provides an overview of their specialties; a hacker(s) specializing in ransomware will use terms such as “crypter” and “locker” more than hackers focusing on SQL injections, who use words such as “SQL” or “injection.”

Once the GCAE receives the adjacency and feature matrices, it encodes them into one low-dimensional embedding. The standard autoencoder’s encoder operation, $h(x) = \sigma(Wx + b)$, is extended to incorporate graph convolution procedures with the following:

$$h(x) = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H W + b)$$

Where σ is the activation function, ReLU. H is the row-wise embedding of the graph nodes in a layer, W is the weight matrix, and b is the bias term. As with the original GCN, the inputted adjacency matrix A is added to an identity matrix I_N to avoid any information loss. The resulting matrix, \tilde{A} , is used with the diagonal degree matrix to execute a symmetric normalization, $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, to avoid any scaling issues. Each layer in the encoder applies these operations while simultaneously reducing the dimensions on the previous layer’s output. This process continues until the desired condensed dimensionality is reached. At this point, the GCAE decoder begins reconstructing the input by extending the standard autoencoder decoder

function of $\sigma(W^*(h(x)) + b)$ (where $h(x) = \sigma(Wx + b)$, or the embedding generated by the encoder) to:

$$\hat{x} = \sigma((\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})(W^*(h(x)) + b))$$

Where σ is the activation function, $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the symmetric normalization, W^* is the weight matrix $h(x)$ is the embedding generated from the encoder (i.e., $h(x) = (\sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}HW + b))$, and b is the bias term. Each decoder layer applies these operations while expanding the dimensionality of the previous layer. Once the input dimensionality is reached, the Mean Squared Error (MSE) between the input x and the reconstructed output \hat{x} is calculated with:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x - \hat{x})^2$$

The resulting error is backpropagated through the GCAE to adjust the weights accordingly. The process encoding, decoding, calculating MSE, and backpropagation continues until MSE is minimized. While all GCAE operations are implemented using the PyTorch, Numpy, Scikit-learn, and Networkx packages in Python on a single Ubuntu machine (4 GB RAM, 50 GB SSD), Algorithm 1 (shown below) summarizes the steps described above in a manner which can be implemented in other software and hardware setups.

ALGORITHM 1. Graph Convolutional Autoencoder (GCAE)

Input: # of nodes N , # of features F , adjacency matrix A ($N \times N$), feature matrix X ($N \times F$), # of encoder layers d_e , # of decoder layers d_d

Output: embedding matrix for each node $E = [e_1, e_2, \dots, e_N]$, where e_i is the embedding for node i

Procedure

$$\tilde{A} = A + I_N, \tilde{D} = \text{degree_matrix}(\tilde{A})$$

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \text{ // normalize the degree matrix}$$

loop:

$$H_1^{(e)} = X$$

foreach GCAE encoder layer $(H_i^{(e)}, W_i^{(e)})$ ($i = 1, 2, \dots, d_e - 1$):

$$H_{i+1}^{(e)} = \text{ReLU}(\hat{A} H_i^{(e)} W_i^{(e)}) \text{ // graph convolution operations for encoder}$$

$E = H_{d_e}^{(e)}$ // embedding generated by encoder

$$H_1^{(d)} = \text{ReLU}(\hat{A} H_{d_e}^{(e)} W_{d_e}^{(e)})$$

foreach GCAE decoder layer $(H_j^{(d)}, W_j^{(d)})$ ($j = 2, 3, \dots, d_d - 1$):

$$H_{i+1}^{(d)} = \text{ReLU}(\hat{A} H_i^{(d)} W_i^{(d)}) \text{ // graph convolution operations for decoder}$$

$$\hat{X} = H_{d_d}^{(d)} \text{ // reconstructed output}$$

$err = \text{MSE}(X, \hat{X})$ // calculate error between reconstructed output and original input

backpropagate err to update $W_i^{(e)}, W_j^{(d)}$, ($i = 1, 2, \dots, d_e; j = 1, 2, \dots, d_d$)

until err is minimized

output E

4.4.4. GCAE Evaluations

As detailed in the previous subsection, GCAE's goal is to create generalized embeddings encompassing nodal and structural features. Consistent with prior literature proposing novel community detection approaches, we evaluate GCAE's effectiveness through a series of benchmark experiments on a gold-standard network. Constructed using one forum in our collection, this network contains 150 nodes separated into three non-overlapping exploit-sharing communities: crypters, keyloggers, and DDoS. This size and community number is consistent with other gold-standard community detection datasets (e.g., Karate). Communities were created by carefully examining the each node's structure and attributes (i.e., post content).

The constructed network is used as a testbed for two experiments that evaluate the performance of GCAE and its components. Both experiments are summarized in Table 30.

Experiment	Justification	Methods	Description	References	Evaluation
Baseline Embeddings	Standard benchmarks	DeepWalk	Random walk and Skip-Gram approaches to create embeddings	Perozzi et al. 2014	Normalized Mutual Information (NMI), homogeneity, completeness, and v-measure; paired t-tests to evaluate statistically significant differences ($p < 0.05$)
		GraRep	Integrates global structural features in each node's embedding	Cao et al. 2015	
		Node2Vec	Extension of DeepWalk that employs biased random walks	Grover and Leskovec 2016	
		LINE	Creates embedding by optimizing an objective function to preserve local first and second order approximations	Tang et al. 2015	
		GCAE without nodal attributes	3 layer GCAE using just structural features	-	
Autoencoder architecture (evaluated at 3, 5, and 7 layers)	Identify ideal architecture to generate the most robust, generalized nodal embeddings	Denoising	Corrupts input data by injecting Gaussian noise	Goodfellow et al. 2016	
		Sparse	Restricts the number of active nodes in the training process		

Table 30. Summary of Benchmark GCAE Experiments

Experiment 1 evaluates the assumption that integrating nodal attributes enhances community detection performance by benchmarking the GCAE against four state-of-the-art

methods that generate node embeddings only using structural features: DeepWalk, Graph Representations (GraRep), Large-scale Information Network Embedding (LINE), and node2vec. DeepWalk employs random walk and Skip-Gram approaches to learn node embeddings (Perozzi et al. 2014). Node2vec is an extension of DeepWalk that replaces the pure random walk with a biased one to create node embeddings (Grover and Leskovec 2016). LINE optimizes an objective function that preserves local and global network structures by capturing first and second order approximations of each node (Tang et al. 2015). GraRep integrates global structural features of the graph into each node’s embedding (Cao et al. 2015). In addition to these algorithms, we also evaluate the GCAE without using nodal attributes.

Experiment 2 identifies how changes in the core autoencoder architecture affects embedding quality. A common practice when proposing a novel deep learning architecture is to evaluate its design against alternative structures (Goodfellow et al. 2016). This can be variations to the core architecture and/or the depth of the proposed network (i.e., number of layers). As presented in Figure 25, the GCAE utilizes the standard autoencoder architecture. Experiment 2 evaluates the performance of the standard GCAE against two alternate autoencoder designs: the sparse and denoising GCAE. Unlike the standard autoencoder where all nodes are active in the training process, the sparse GCAE imposes a sparsity constraint on the weights that restricts the amount of active nodes. The denoising GCAE keeps all nodes active, but injects Gaussian noise into the input data. While other studies using autoencoders have shown that sparsity and denoising constraints force the autoencoder to learn more robust representations of the input data, it is unknown whether the same holds true for the GCAE. Consistent with standard experimental setups, we evaluate the performance of all three GCAE’s using three, five, and seven layers.

In both experiments, embeddings generated from the GCAE and benchmark approaches are inputted into k-means (300 iterations) for community detection. Four well-established metrics evaluate community detection performance: Normalized Mutual Information (NMI), homogeneity, completeness, and V-measure. Each metric evaluates community detection performance on slightly different criteria by calculating a scalar value between 0.0 (random assignment) and 1.0 (perfect match). NMI measures cluster quality by evaluating the mutually shared information (i.e., the number of nodes with the same label) between clusters. Homogeneity measures whether each cluster contains only nodes from the same community. Completeness identifies whether nodes within the same community in the gold-standard set appear in the same cluster. V-Measure calculates the harmonic mean of homogeneity and completeness (Rosenberg and Hirschberg 2007). In both experiments, paired t-tests were used to identify statistically significant differences between methods. Differences were considered significant if the p-value was lower than 0.05.

4.4.5. Key Hacker Identification and Community Detection: Case Study

The last component of our framework illustrates the practical utility of the GCAE and EDC with an in-depth case study of an English hacker forum with 137,664 posts (2,720 containing exploits) made by 7,011 hackers between 2/07/2005 and 10/22/2017. This case study follows the steps a CTI professional or law enforcement official can take when aiming to get an in-depth perspective on an exploit sharing hacker forum. First, we construct a network as described in earlier: nodes represent all hackers within a forum thread containing an exploit and edges denote links from a hacker sharing an exploit to all other hackers in a thread receiving the exploit. This formulation captures all hackers in the exploit sharing community, but omits those engaging in general forum chatter. After network construction, an array of

descriptive statistics are calculated to provide a detailed overview of the overall network dynamics. Following these computations, the GCAE is applied to the network to generate embeddings for each node based on its structural and nodal attributes. The generated embeddings are inputted into k-means to identify communities. The EDC is used to pinpoint key hackers both within the detected communities and the overall network.

4.5. Results and Discussion: Experiments and Case Study

4.5.1. Experiment 1: GCAE vs Structural Embeddings

Experiment 1 evaluated the GCAE against unsupervised state-of-the-art embedding approaches that only account for structural features: DeepWalk, Node2Vec, LINE, GraRep, and the GCAE without nodal attributes. For baseline approaches, we used the default hyperparameter settings as provided by the OpenNE Python package. Both GCAEs used three layer architectures. All methods extracted 128 dimensional embeddings. Full evaluation results are summarized in Table 31 and visualized in Figure 26.

Embedding Approach	NMI	Homogeneity	Completeness	V-Measure
GCAE (3 layer)	57.81%	51.78%	65.57%	57.86%
DeepWalk	44.97%*	41.45%*	49.03%*	44.92%*
Node2Vec	44.51%*	42.72%*	46.43%	44.50%*
GCAE (3 layer, without features)	40.75%*	34.80%*	54.13%*	42.36%*
LINE	22.36%*	22.36%*	23.35%	22.84%*
GraRep	22.31%*	21.50%*	23.56%	22.48%*

Table 31. GCAE Benchmark Evaluation Results
(* indicates statistically significant difference at $p < 0.05$)

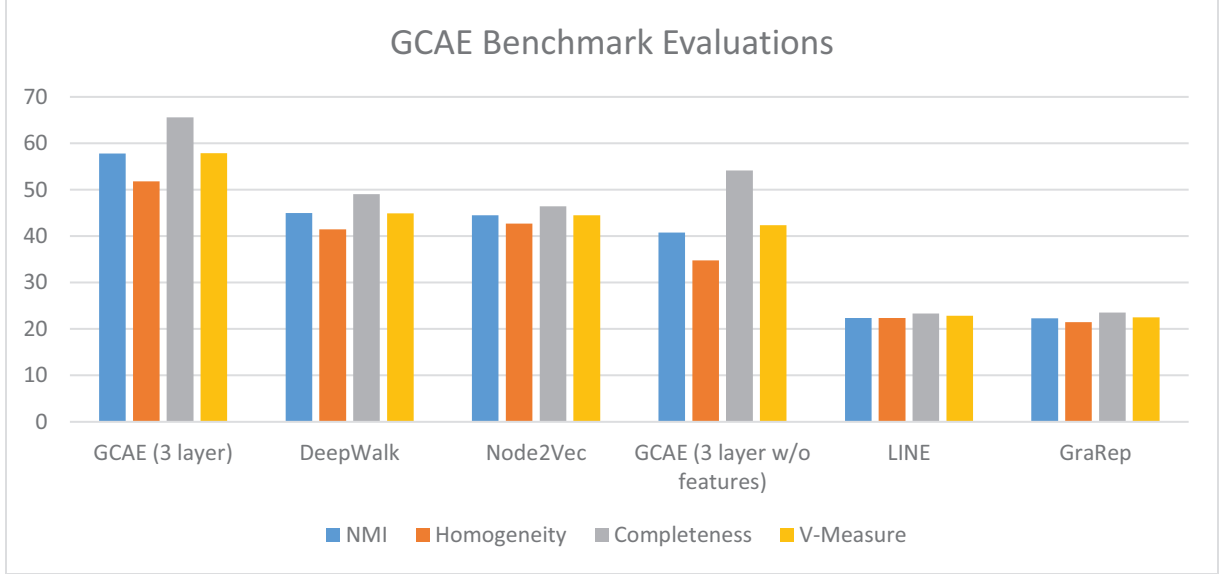


Figure 26. GCAE Benchmark Evaluation Results

Overall, the GCAE outperformed all approaches in terms of NMI, homogeneity, completeness, and V-Measure by statistically significant margins ($p < 0.05$). The random walk based node2vec and DeepWalk had similar performances in terms of V-measure (harmonic mean of homogeneity and completeness), with 44.50% and 42.36% respectively. The GCAE without nodal attributes achieved a comparable V-measure of 42.36%, a score which outperformed both LINE (22.84%) and GraRep (22.48%).

GCAE’s strong performance indicates that incorporating nodal attributes significantly enhance community detection. From a domain perspective, this means that nodes with limited structural information, but still containing valuable nodal attributes, can be effectively grouped into the appropriate clusters. This is particularly useful when a node has just joined the network (e.g., experienced hacker from another forum), has posted content, but has not yet communicated with many other hackers. GCAE’s strong performance also indicates that both structural and nodal attributes can be effectively represented within the same length embedding (128 dimensions) as other methods. While the black-box nature of deep learning prevents us from directly identifying which features were included in this embedding, this result illustrates

GCAE’s ability to distill the most important attributes from two feature sets (structural and nodal).

4.5.2. Experiment 2: GCAE vs Sparse and Denoising GCAE

Experiment 2 evaluated GCAE’s core architecture against two alternatives, the sparse and denoising GCAEs, at varying levels of depth. To ensure consistency of comparisons, we trained each GCAE architecture for 200 epochs (i.e., 200 passes over the data), used a learning rate of 0.01, imposed a dropout of 0.5, and used the ReLU activation function. Experiment results indicate that the GCAE outperforms both the denoising and sparse variations. Full results are summarized in Table 32.

GCAE Depth	GCAE Configuration	NMI	Homogeneity	Completeness	V-Measure
3 layer	GCAE	57.81%	51.78%	65.57%	57.86%
	Sparse GCAE	51.17%	46.00%	58.00%	51.37%
	Denoising GCAE	54.08%	52.41%	55.89%	54.09%
5 layer	GCAE	49.46%*	49.12%*	49.80%*	49.46%*
	Sparse GCAE	42.76%*	41.63%*	45.68%*	43.56%*
	Denoising GCAE	44.09%*	44.58%*	46.90%*	45.71%*
7 layer	GCAE	39.81%*	38.79%*	40.94%*	39.84%*
	Sparse GCAE	22.36%*	21.67%*	23.40%*	22.50%*
	Denoising GCAE	26.75%*	26.71%*	26.79%*	26.75%*

Table 32. GCAE Architecture Evaluation Results
 (* indicates statistically significant difference at $p < 0.05$)

Experiment 2 reveals two insights that can guide future researchers in their GCAE architecture selection. First, the standard GCAE outperformed the sparse and denoising variations in terms of NMI, homogeneity, completeness, and V-Measure. This finding was consistent irrespective of GCAE depth. These results indicate that unlike past deep learning studies employing autoencoders, incorporating sparsity and Gaussian noise does not create more robust embeddings, but hurts it. An explanation for this performance degradation is the graph convolution operations employed by the GCAE. Contrast to the standard autoencoder

that uses relatively simple layer-wise propagations, graph convolution operations employ more sophisticated computations (e.g., adjacency matrix normalization). Constraining the activations of nodes within the encoder (sparse GCAE), or injecting noise into the input data (denoising GCAE) hampers GCAEs ability to effectively apply these computations. When examining the two alternative architectures, the denoising GCAE outperformed the sparse GCAE across all levels of depth. This reveals that enforcing sparsity constraints has more detrimental effects than the insertion of Gaussian noise.

The second key insight from experiment 2 is the inverse relationship of GCAE depth and performance for all GCAE variations. This contrasts with other studies, which have found that additional depth boosts overall results (Goodfellow et al. 2016). Examining the data input directly can shed light on this behavior. Unlike the standard autoencoder which aims to learn a single two dimensional feature matrix with n rows and m columns, the GCAE receives two matrices as inputs: adjacency and feature. It is likely that the encoder in deeper GCAEs over-summarizes the data by steadily removing the most important features either or both feature sets. Conversely, shallower architectures will learn the most salient representation directly, allowing the decoder to effectively reconstruct the input. Given these findings, the case study (results in following subsection) is executed with a three layer GCAE without sparsity constraints or Gaussian noise.

4.5.3. Case Study Results

Following network construction for our case study, we ran a series of network and node level descriptive statistics. Network level metrics include number of nodes, edges, network diameter, graph density, average path length, and average clustering coefficient. Each provides insight into the manner and speed at which hackers communicate with each other. Node level

metrics are the minimum, maximum, and average degree and EDC scores. Table 33 summarizes the results of both sets of calculations.

Metric Category	Metric	Value
Network Level Metrics	# of nodes	2,911
	# of edges	11,426
	Network diameter	9
	Graph density	0.001
	Average path length	3.405
	Average clustering coefficient	0.157
Node Level Metrics	Minimum degree	1
	Maximum degree	512
	Average degree	7.796
	Minimum EDC	0.118
	Maximum EDC	117.126
	Average EDC	1.301

Table 33. Topological and Node Level Descriptive Statistics

Out of the possible 7,011 hackers, 2,911/7,011 (41.5%) participate in exploit sharing (consuming and/or contributing) activities. The network diameter of nine indicates that the overall network is compact in nature, thus allowing hackers to take a minimal number of steps to reach others (average path length of 3.405). Despite the opportunity to interact with many hackers quickly, the low graph density of 0.001 suggests that many participants opt to interact with a select few hackers. This preference is reflected in the degree distribution. Both the standard degree (minimum 1, maximum 512, average 7.796) and the EDC (minimum 0.118, maximum 117.126, average 1.301) follow a power-law distribution, where a few hackers possess high centrality scores and act as hubs within the network. We summarize the top 10 hackers based on their EDC in Table 34. For each hacker, we provide their EDC, degree, and betweenness centralities. All three are strong indicators of key members within the network. The EDC and degree are decomposed into in and out measures, as doing so provides insight into each hacker's skill level. We also detail each hacker's forum role. To prevent being targeted by the listed members, we list each hacker's ID rather than their screen name.

Hacker		EDC				Degree				Betweenness	
ID	Status	In	Out	Total	Rank*	In	Out	Total	Rank	Value	Rank
1982	Sr. Member	17.215	99.911	117.127	1	78	345	423	3	173,758.75	4
2193	Sr. Member	24.850	90.264	115.114	2	117	395	512	1	304,448.20	1
1	Admin	20.328	91.782	112.110	3	103	392	495	2	248,763.40	2
5797	Sr. Member	26.699	63.794	90.494	4	109	252	361	4	229,107.90	3
3139	Sr. Member	45.363	15.591	60.954	5	161	76	237	6	109,747.20	6
2605	Sr. Member	21.129	34.302	55.431	6	109	154	263	5	149,321.70	5
87	Sr. Member	18.760	23.518	42.278	7	84	122	206	8	80,857.04	9
2027	Sr. Member	18.138	22.743	40.882	8	94	142	236	7	106,824.40	7
7320	Sr. Member	6.041	29.939	35.980	9	34	146	180	10	81,677.69	8
641	Admin	26.012	9.641	35.653	10	117	65	182	9	59,410.03	12

Table 34. Ranking of Top Hackers in Constructed Network
 (* rank based on total EDC)

The top ten hackers are senior members or administrators within the forum. This standing may explain why other hackers are partial to interacting with a selected set of members. Among the top ten, hackers 3139 at rank five and 641 at rank ten consume more exploits than they provide to the community. This indicates that while ranked in the top tier of hackers, they possess a lower skill level than other hackers due to their propensity to consume exploits made by others rather than developing and sharing their own. These characteristics can help law enforcement prioritize these individuals lower than those who primarily contribute exploits (e.g., 1982, 1) when executing investigations.

Incorporating the time of each hacker's exploit consumption/contributions provides deeper intelligence. For example, the standard degree would have ranked hacker 1982 number three overall. However, using the EDC, hacker 1982 tops the list. While this difference may be negligible in other contexts, CTI professionals and law enforcement work in an environment where accuracy and granularity of intelligence is of the utmost importance. Each hacker investigation requires significant time, expertise, and financial investment. These resource demands limits the number of pursuits that can be executed. Clearly and accurately understanding current exploit sharing hackers can help professionals precisely target network members for surveillance and/or removal in a timely fashion. Moreover, it can suggest investigative leads that would otherwise have been overlooked or received a lower priority level (e.g., hacker 1982).

As indicated in previous sections, CTI professionals are not only interested in identifying key hackers in the overall network, but also groups of hackers based on their attack preferences (e.g., SQL injection, ransomware, etc.). This enables the generation of intelligence most relevant to the critical assets they are trying to protect within their organization. To this

end, we apply the GCAE to the generated network to create embeddings for each node, and input the embeddings into k-means to extract communities. To ensure we avoided extracting coarse communities with numerous sub-specialties, we ran k-means to extract six clusters to pinpoint finer grained groups. Community detection results are visually depicted in Figure 27.

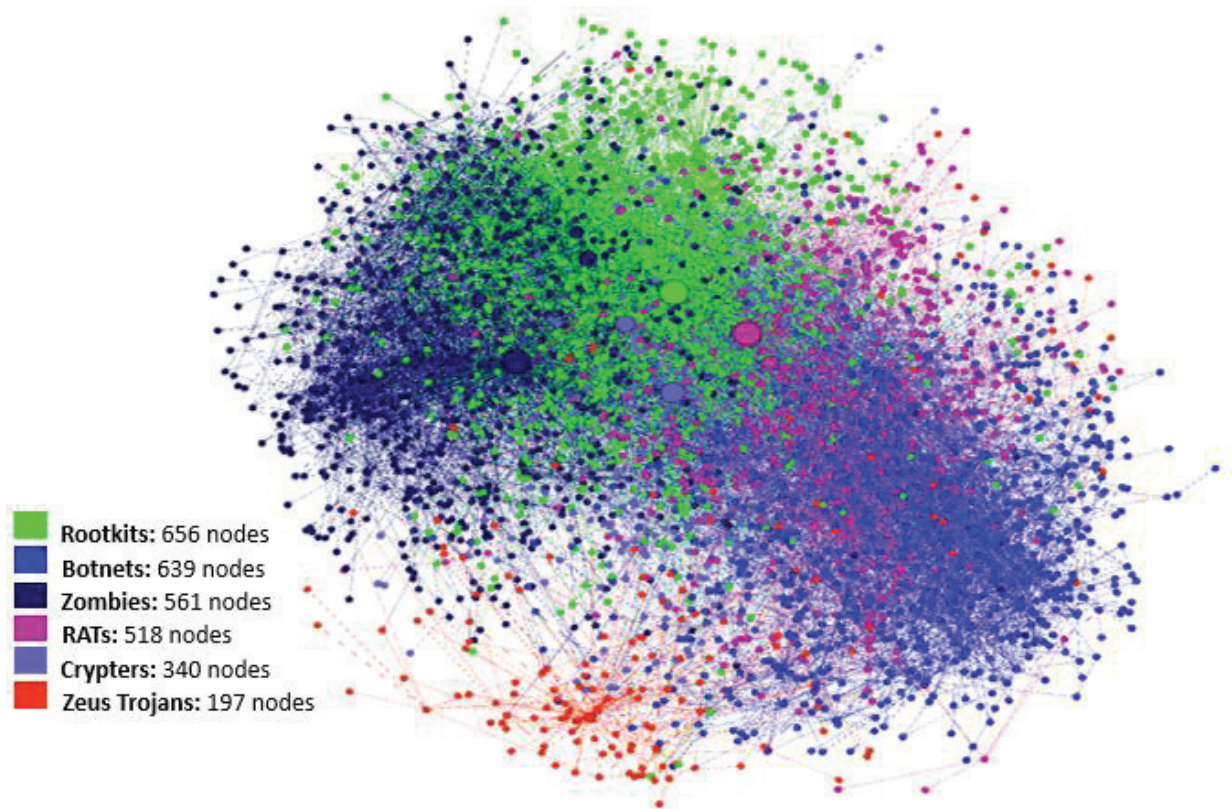


Figure 27. Communities Identified Using GCAE Embeddings

We manually explored selected nodes within each identified community to understand its specialties. The six communities breaks down as follows. 656 nodes (light green, top middle) focus on sharing rootkits, malware designed to gain root access on a machine to control low level system processes such as memory dumps, caches, etc. 639 nodes (light blue, bottom right) specialize in botnet software (e.g., Command and Control), technologies often used to conduct DDoS. 561 (dark blue, upper left) concentrate on providing zombie services designed to use computing resources on unsuspecting victim machines to execute malicious activities.

518 (magenta, center right) focus on Remote Administration Trojans (RATs), tools designed to gain high-level command over a system to control and monitor its processes and peripherals (e.g., webcam). 340 nodes (light purple, center) engage primarily in sharing crypters, a core technology for ransomware. The remaining 197 nodes (red, bottom) specialize in Zeus Trojans, exploits designed specifically for the systems within financial institutions. Interested stakeholders and law enforcement can also delve deeper into the key hackers in each community (illustrated in Table 35).

Community (Color)	# of Nodes	Top Two Hackers	Top Ten Overall?	In-EDC	Out-EDC	Total EDC	Forum Role
Rootkits (Light Green)	656	1982	Yes, 1	17.215	99.911	117.127	Sr. Member
		2605	Yes, 6	21.128	34.302	55.431	Sr. Member
Botnets (Light Blue)	639	7320	Yes, 9	6.041	29.939	35.980	Sr. Member
		6309	No	5.230	16.153	21.383	Sr. Member
Zombies (Dark Blue)	561	1	Yes, 3	20.328	91.782	112.110	Admin
		87	Yes, 7	18.759	23.519	42.278	Sr. Member
RATs (Magenta)	518	2193	Yes, 2	24.850	90.264	115.114	Sr. Member
		2027	Yes, 8	18.138	22.743	40.882	Sr. Member
Crypters (Light Purple)	340	5797	Yes, 4	26.699	63.794	90.494	Sr. Member
		3139	Yes, 5	45.363	15.591	60.954	Sr. Member
Zeus Exploits (red)	197	16000	No	1.208	21.920	23.128	Sr. Member
		33038	No	0	1.562	1.562	Jr. Member

Table 35. Each Community's Top Hackers (Ranked by EDC)

Nine of the top ten hackers depicted in Table 34 (earlier) are also key members within their respective communities (Table 35). Our explorations of the constructed network indicate that these top hackers largely specialize in one area; they do not provide or consume exploits outside of their areas of expertise. A possible explanation for this is their desire to gain significant reputation for a specific area by consistently providing novel, advanced tools and techniques. CTI professionals can use this knowledge to delve deeper into the group of hackers and the key hacker(s) most pertinent to them. For example, healthcare organizations afflicted with ransomware can examine hacker 5797 (total EDC 60.954) to identify the new crypter

technologies they provide. Likewise, financial institutions investigate hacker 16000 (not listed in the top ten overall) to understand the characteristics of their Zeus exploits and proactively formulate the appropriate cyber-defenses against them.

4.6. Conclusion and Future Directions

Cybersecurity is a grand societal challenge. CTI has emerged as a valuable approach to combat the increasing threats of malicious hackers. Despite CTI's promise, existing practices rely heavily on internal network data, leading to inherently reactive intelligence. As a result, numerous cybersecurity experts have recognized that cyber-attacks remain on an unfortunate uptick. The vast, ever-evolving online hacker community can provide a novel data source that can aid in the proactive identification of malicious hackers, and in turn, reduce the ever-growing concern of cyber-attacks.

In this study, we explore the relatively untapped online hacker forums for two purposes: to accurately identify current key exploit sharing hackers and communities of exploit sharing hackers. To achieve the first goal, we developed the EDC, a measure that incorporates the age of exploits provided by a hacker. To execute the second goal, we designed a novel deep learning architecture, the GCAE. The GCAE extends the standard autoencoder to incorporate graph convolutional operations to generate a generalized node embedding accounting for a node's structural and nodal attributes. Rigorous evaluation of the GCAE indicates that using both nodal and structural features outperforms state-of-the-art methods using only one in community detection tasks.

The practical utility of both the EDC and GCAE were illustrated with an in-depth case study, where we identified six communities of hackers sharing exploits such as crypters, Zeus malware, botnets, and others. The top two hackers in terms of the recency of their exploit

contributions were pinpointed to aid CTI professionals and law enforcement in executing targeted investigations. While demonstrated in the context of hacker forums, the GCAE can add value to generate embeddings in other node-attributed networks in social media contexts.

Several extensions can be made to this research to offer additional value to CTI professionals and law enforcement officials. First, future work can examine how communities are born, develop, and die. Along these lines, additional work can be done to predict links between hackers and pinpoint potentially dangerous hackers in the early stages of their development. Future studies can also examine how exploits disseminate within communities. Executing each of research avenues can provide novel insights that can help develop a safer and more secure society.

5. ESSAY IV: IDENTIFYING EMERGING HACKER EXPLOITS: A DIACHRONIC GRAPH CONVOLUTIONAL AUTOENCODER FRAMEWORK

5.1. 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 using sophisticated exploits. Cybersecurity experts have appraised the total cost of hacktivism, espionage, cyberwarfare, and other hacking activities against major entities such as Equifax, Uber, and Yahoo! at \$450 billion annually. To combat this dire issue, many organizations have started developing and using Cyber Threat Intelligence (CTI).

At its core, CTI is a data-driven process that focuses on developing timely, relevant, and actionable intelligence about emerging threats and key threat actors (i.e., hackers) to enable effective cybersecurity decisions. 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 exploits used against the networks. Despite the maturity and value of these processes, the derived intelligence is inherently reactive to known exploits, as the data analyzed are past network events. As a result, CTI professionals from the acclaimed SANS Institute have recently noted that “most organizations are still reactive to alerts and incidents instead of proactively seeking out the threats” (Lee and Lee

2017). Consequently, the quantity, severity, and sophistication of exploits used in cyber-attacks increase annually.

To combat these concerns, experts have suggested proactively examine emerging exploits in the vast, international online hacker community (Bromiley 2016; Shackleford 2016). 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. Although each has CTI value, hacker forums are particularly unique value to CTI experts. Unlike other platforms, forums allow hackers to freely share and discuss malicious cyber-attack exploits. Figure 28 illustrates one example, where a hacker provides Bitcoin miner 0-day exploit for other hackers to freely download and use.

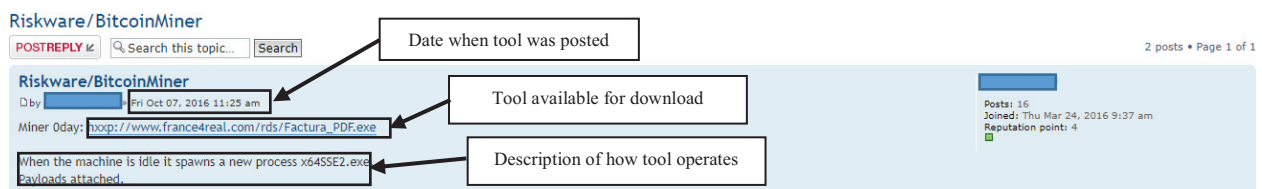


Figure 28. Example of a hacker providing a Bitcoin Miner 0-day exploit

Hackers have obtained exploits in hacker forums to execute well-known breaches. One notable example is the Target incident, where hackers procured the BlackPOS malware months before executing the attack. The severity of this event motivates the careful analysis of hacker forum data to identify emerging exploits. However, hacker forums contain hundreds of thousands of unstructured, un-sanitized text records. Hackers rapidly evolve in their skill-sets; thus, they develop new malware and augment existing exploits with novel functions. Compounding this issue is the unclear semantics of hacker terminology (e.g., injection can refer to memory, SQL, or process), and how they shift over time. Existing CTI analytics are ill-equipped for these unique characteristics. Moreover, traditional and emerging text analytics

approaches employed in extant hacker forum literature require significant extensions to generate valuable CTI. The combination of these challenges present numerous challenges for CTI professionals and motivate the development of innovative CTI text analytics.

In this study, we aim to develop proactive CTI capabilities by exploring online hacker forums to identify emerging exploit trends in terms of popularity and tool functionality. To achieve these goals, we create a novel deep learning architecture, the Diachronic Graph Convolutional Autoencoder (D-GCAE). The D-GCAE customizes the core operations of an emerging method, the Graph Convolutional Network (GCN), and integrates them into the autoencoder architecture. This novel method operates on a Graph-of-Words (GoW) representation of hacker forum exploit text to generate low-dimensional word embeddings in an unsupervised manner. Semantic displacement measures adopted from diachronic linguistics literature identify how exploit terminology evolves over time. Rigorous evaluation of the D-GCAE demonstrates its ability to generate higher quality embeddings than state-of-the-art approaches for selected downstream tasks. Although the D-GCAE’s practical utility is illustrated with an in-depth ransomware case study, the proposed approach can benefit multiple social media and/or network science applications.

The remainder of this essay is organized as follows. First, we review literature related to the online hacker community, text graphs, GCNs, autoencoders, and diachronic word embeddings. Second, we identify methodological and domain research gaps within existing studies and posit questions for study. Third, we summarize our research design, testbed, and experimental procedure. Subsequently, we summarize our evaluation and case study results. We conclude this work by highlighting our contributions and identifying promising directions for future research.

5.2. Literature Review

Five areas of literature are searched and examined: (1) hacker community research to identify key forum features and past efforts detecting emerging exploit trends, (2) text graphs to review approaches and benefits of representing text as a network, (3) GCNs to understand its core operations and applications on text graphs, (4) autoencoders to guide the D-GCAE's architecture, and (5) diachronic word embeddings to identify approaches to compare embeddings for temporal data.

5.2.1. *Hacker Community Research*

As mentioned in the introduction, hackers from major geo-political regions use four major online platforms to share malicious tools and knowledge: DarkNet Marketplaces, forums, carding shops, and IRC channels (Benjamin et al. 2015; Samtani et al. 2017). Among these, hacker forums are particularly valuable for detecting emerging exploits. Carding shops and IRC platforms do not provide the mechanisms for hackers to freely share exploits, while DarkNet Marketplaces have more drug, pornography, and weapon material than cybersecurity related content. Forums also provide richer data metadata and characteristics, namely exploit post date and extensive exploit post content (e.g., Figure 28), features not consistently available in other platforms. These characteristics have motivated numerous researchers to identify hacker forum exploits. Table 36 summarizes selected recent studies.

Year	Author	Data Source	Analytics*	Identified Exploits	Exploit Trend Identification?	Trend Identification Method
2018	Sapienza et al.	Twitter, forums	Keyword approach	Botnets, DDoS	Yes	Exploit term frequency
2017	Samtani et al.	Eight forums	LDA and SVM	RAT's crypters, keyloggers, botnets, DDoS, SQLi, web exploits	Yes	Exploit term frequency
2017	Grisham et al.	Four forums	RNN	Mobile malware	Yes	Exploit term frequency
2016	Li et al.	Three forums	sLDA	Malware, phishing, botnets	No	None
2016	Nunes et al.	21 forums	SVM	Botnets, keyloggers, worms, 0-days, Android malware	No	None
2016	Zhao et al.	12 forums	Manual	DDoS, web exploits	No	None
2016	Samtani et al.	Two forums	LDA and SVM	Bots, crypters, keyloggers, phishing, SQL injection, bank exploits, XSS, Zeus	Yes	Exploit term frequency
2014	Hutchings and Holt	13 forums	Manual	Keyloggers, phishing, banking Trojans	No	None

Table 36. Selected Studies Identifying Exploits in Online Hacker Forums

***Note: LDA=Latent Dirichlet Allocation; RNN=Recurrent Neural Network; sLDA= supervised Latent Dirichlet Allocation; SVM=Support Vector Machine**

Scholars have employed support vector machine (SVM), topic modeling, keyword approaches, and interviews with subject matter experts (SMEs) to identify exploits in forums (Hutchings and Holt 2014; Samtani et al. 2015; Samtani et al. 2016; Zhao et al. 2016). Analysis reveals that hackers freely share exploits such as botnets, email hacks, exploit kits, keyloggers, web exploits, remote administration tools (RAT's), bank exploits, denial of service (DoS), and many others. Several studies have gone one step beyond identifying exploits to detecting the overall exploit trends. For example, Grisham et al. (2017) plotted the number of mobile malware occurrences within a major Arabic hacker forum. Other studies have monitored the frequency of exploit terms (e.g., “botnet,” “crypter,” etc.) over a selected time period (Samtani et al. 2016; Samtani et al. 2017; Sapienza et al. 2018).

While providing CTI value, using term frequencies to identify exploit trends has 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 constantly expand their exploit vocabulary. As a result, new exploit names can be initially overlooked. One example is Mirai, the botnet which executed the 2016 Internet DoS. Although appearing months prior to the attack, it remained undetected as cybersecurity professionals were unaware of the new botnet terminology. Finally, term frequencies cannot capture the distance or relationships of a term with others. Thus, it is unclear how exploit terminology evolves in usage over time (i.e., semantic shifts). These limitations necessitate an alternative approach to represent hacker forum text. One method gaining traction within the Natural Language Processing (NLP) community are text graphs. Text graphs reveal relationships, patterns, and regularities within a corpora not captured in standard representations (e.g., bag of words).

While currently applied in other social media contexts, they can offer significant value in modeling hacker exploit text in a novel fashion.

5.2.2. Text Graphs

Text graphs build upon network science principles to represent text in a graph. Like other network science applications, text graphs are constructed with two building blocks: nodes and edges. Nodes are text units such as words, sentences, collocations, word senses, 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) (Nastase et al. 2015). Nodes in this network are words, and edges indicate whether two words appear together in a text unit (e.g., document, sentence, etc.). The edges can be weighted to denote how frequently two words have co-occurred.

All node relationships in a GoW G are held in an $N \times N$ adjacency matrix A , where N denotes the number of nodes (i.e., words) (Barabasi 2016). 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' vocabulary from different perspectives. Network-level metrics (e.g., network density, clustering coefficient, etc.) provide insight into the richness and expressiveness of a corpus' vocabulary. Node-level measures, degree, eigenvector, betweenness, and closeness, pinpoint key words based on different criteria. Count-based methods (degree and eigenvector) sum and/or weight the number of in and out-links from a node. Distance-based methods (betweenness and closeness) measure how close two nodes are.

Scholars have integrated selected measures as features to achieve state-of-the-art performances in authorship analysis (Akimushkin et al. 2017) and text classification (Rousseau et al. 2015). Recent years, however, has seen a shift from manual feature engineering (often ad-hoc, labor and time-intensive) to automatic generation of latent, low-dimensional representations via deep learning. Deep learning is a category of machine learning algorithms that employ neural networks with multiple layers of non-linear computations to learn features from a data input (Goodfellow et al. 2016). Deep learning has enabled remarkable performances across numerous applications without manual feature selection. In the context of hacker forum GoW's, deep learning can reveal latent local and global relationships of exploit terms that would otherwise be overlooked. One emerging deep learning approach primarily operating on text graphs is the GCN (Graph Convolutional Network), which we review next.

5.2.3. Graph Convolutional Networks (GCNs)

Developed in 2017, GCNs learn embeddings from node-attributed networks for supervised learning tasks (Kipf and Welling 2017). The GCN learns a function $f(X, A)$, where X is an $N \times N$ (i.e., each node's feature vector, if available) and A is the $N \times N$ adjacency matrix from the graph G . The GCN incorporates neural network operations to generate an $N \times F$ embedding matrix, Z . N is the number of nodes from the graph G , and F is the length of the embedding (i.e., feature vector) for each node. The GCN layerwise propagation rule is as follows:

$$H^{l+1} = \sigma(AH^{(l)}W^{(l)})$$

Where σ is the neural network activation function (most commonly Rectified Linear Unit (ReLU)), A is G 's adjacency matrix, $H^{(l)}$ is the row-wise embedding of the graph nodes in the l th layer, and $W^{(l)}$ is the weight parameter matrix. This formulation controls the number

of parameters used by the neural network and is also robust network size shifts. However, two issues arise. First, the formulation accounts for each node's neighbors but does not incorporate each node's relationship to itself (i.e., self-loops). This results in some information loss. To address this issue, an identity matrix I_N is added to A to create \tilde{A} . Second, multiplying by A can result scaling issues in the output embedding. To address this, a symmetric normalization $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is created, where D is the node degree matrix. Both adjustments create a generalized GCN propagation rule:

$$H^{l+1} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

Where \hat{A} is $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. This computation is a first-order approximation of spectral graph convolution, where it convolves around each node's neighborhood to capture a rich set of information (see Kipf and Welling (2017) for full proof). As a result, it simulates the Convolutional Neural Network's (CNN's) convolution and pooling procedures. GCNs' robustness has enabled scholars to pursue numerous research inquiries on various networks, primarily text graphs. Selected recent work is summarized Table 37.

Year	Author	Dataset(s)	Task	Nodes	Edges
2018	Abu-El-Haija et al.	Pubmed, Citeseer, Cora	Semi-supervised classification	Documents	Citations
2018	Chen et al.	Cora, Pubmed, Reddit	Node classification	Documents	Citations
2017	Kipf et al.	Cora, Citeseer, Pubmed	Link prediction	Documents	Citations
2017	Hamilton et al.	Citation, Reddit, PPI	Node classification	Documents	Citations
2017	Yasunaga et al.	Data Understanding Conferences Papers	Document classification	Sentences	Cosine similarity threshold
2017	Kipf and Welling	Citeseer, Cora, Pubmed, Nell	Semi-supervised classification	Documents	Citations

Table 37. Selected Studies Using Graph Convolutional Networks on Text Graphs

GCNs classified documents classification using semi-supervised (Haija et al. 2018; Kipf and Welling 2017) and fully-supervised (Chen et al. 2018; Hamilton et al. 2017; Yasunaga et al. 2017) methods. Researchers have also applied the GCN for link prediction in document networks (Kipf et al. 2017). Like other supervised algorithms, the GCN needs 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 others (e.g., generating a embedding for a word in GoW). Given our objective, this drawback requires the core GCN operations be adapted into a deep learning architecture that generates task-independent node embeddings without a gold-standard dataset (a resource not available for all corpora). One promising architecture is the autoencoder, which is reviewed next.

5.2.4. Autoencoders

The autoencoder is an unsupervised deep learning algorithm that learns an embedding for a data input (Goodfellow et al. 2016). The autoencoder performs the same feed-forward and backpropagation computations as the standard ANN. Unlike the ANN, the autoencoder has three major components: encoder, compressed embedding, and decoder. Figure 29 depicts a generic, 7-layer autoencoder.

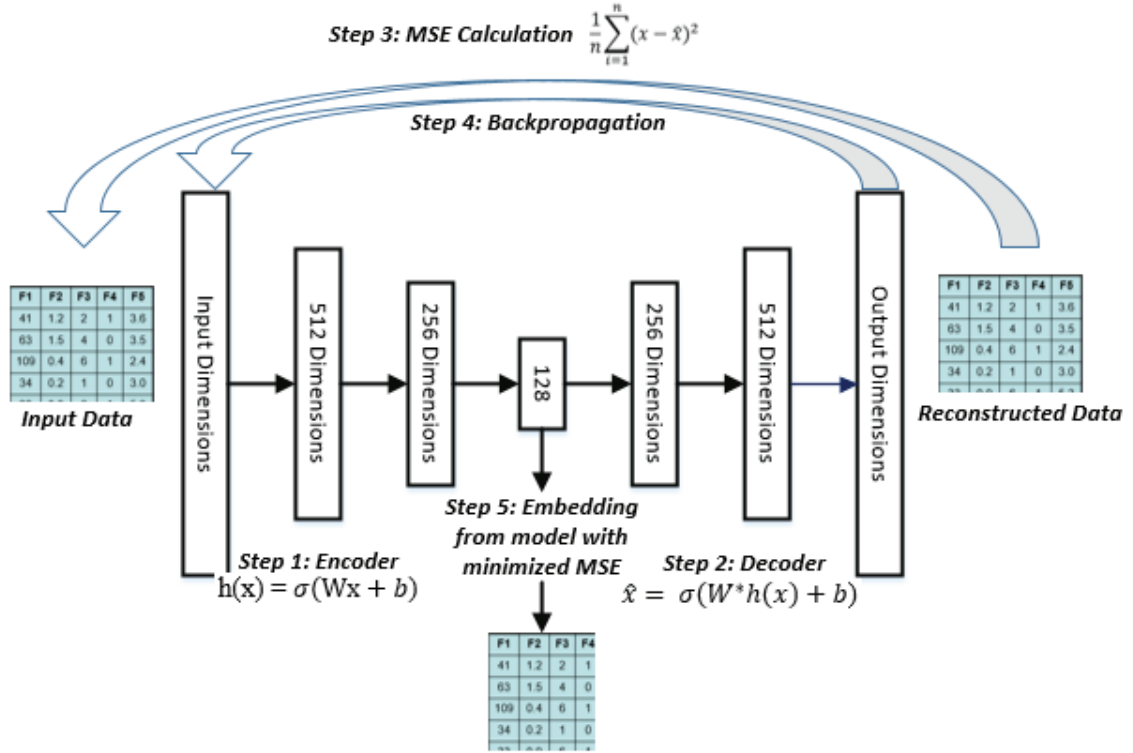


Figure 29. Autoencoder Operations

From left: Step 1 uses a series of non-linear activation layers to reduce dimensionality of the input data until a bottleneck is reached. Step 2 uses the bottlenecked embedding to reconstruct the input at the output layer. Step 3 calculates the Mean Squared Error (MSE) between the reconstructed output and the original input. Step 4 uses backpropagation to adjust the encoder and decoder weights. Steps 1-4 repeat until the MSE between the output and input is minimized.

The encoder receives a data feature matrix x as input and learns a function $h(x) = \sigma(Wx + b)$. σ is the activation function, W is the weight matrix for layer l in the encoder, and b is the bias term. Each layer in the encoder takes the input from the previous layer and reduces its dimensionality. This process continues until a condensed representation (i.e., bottleneck) $h(x)$ is reached (center of Figure 29). $h(x)$ is used as input for the decoder, which learns a function $\hat{x} = \sigma(W^*h(x) + b)$ to reconstruct the original output. Each layer applies these operations while increasing previous layer's dimensionality. This process continues until the output dimensionality matches the original input. At this point, the Mean Squared Error (MSE) between the input x and the reconstructed output \hat{x} is calculated with:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x - \hat{x})^2$$

The computed MSE is backpropagated through the decoder and encoder layers to update the weights. This process is analogous to the standard ANN, where ground-truth labels are compared to the predicted output and the weights are adjusted. Encoding, decoding, error calculation, and weight adjustment continues until the MSE is minimized. The bottleneck in the autoencoder with minimal MSE serves as a general, condensed, latent representation that can be used in subsequent tasks (e.g., clustering, classification).

Despite showing promise in integrating GCN operations, autoencoder’s 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 embeddings across time-spells. Thus, embedding spaces need alignment via an external mechanism to enable fair and accurate comparisons. In this study, adapting the GCN into an autoencoder and operating it on a GoW would result in a novel approach to generating word embeddings. Using this perspective, we can examine an emerging stream of literature from information retrieval and linguistics disciplines that have focused on aligning embedding spaces over multiple time-spells to model semantic shifts of word embeddings generated from popular models (e.g., word2vec). This literature, referred to as diachronic word embeddings, is reviewed next.

5.2.5. Diachronic Word Embeddings

Contrast to synchronic linguistics that studies language at one time point, diachronic linguistics (i.e., historical linguistics) examines language development and evolution. While traditionally reliant on manual approaches, the advent of the synchronic Neural Network Language Model (NNLM) word embedding approach word2vec by Mikolov et al. (2013)

spurred the development of a new area of inquiry: diachronic word embeddings (Hamilton et al. 2016). 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 spell. While word embeddings within time-spells can be compared for semantic similarity, 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, $\mathbf{W}^{(t)} \in \mathbb{R}^{d \times |V|}$, where t is the time-spell (Hamilton et al. 2016). Matrices generated at two time-spells are aligned using orthogonal Procrustes. Specifically, embedding spaces are aligned across time-periods while preserving cosine similarities by optimizing the following:

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

Where $\|\cdot\|_F$ denotes the Frobenius norm. This solution conforms to the best rotational alignment of both embedding spaces and can be efficiently attained by using an application of Singular Value Decomposition (SVD) (Schonemann 1966). Aligning spaces facilitates the computation of a novel linguistic measure to model language evolution: semantic displacement measurement. Semantic displacement identifies a word’s semantic shift across time-periods by measuring the cosine distance of a word at two time-periods (i.e., $\text{cosine-dist}(w_t, w_{t+\Delta})$). 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 changes across the centuries on the Google books corpus (Hamilton et al. 2016a), evolution of English on the Corpus of

Historical American English (Hamilton et al. 2016b), and terminology usage in New York Times articles (Yao et al. 2018).

5.3. Research Gaps and Questions

Our literature review revealed several research gaps. From a domain perspective, existing hacker forum exploit studies rely on term frequencies to detect emerging exploit trends. Despite its value, this approach cannot account for relationships between words; thus, they are unable to identify how hacker exploit semantics evolve over time and what new exploit terms are and pertain to. GoWs ability to capture relationships between text units can offer significant value in modeling hacker exploit content. While GCNs have emerged as a state-of-the-art approach to generate latent node embeddings from graph data (most applications on text graphs, but not on GoW's) they are designed for supervised learning. Consequently, generated embeddings are task specific. Autoencoders show promise in adapting graph convolutions to create task-independent node (i.e., word) embeddings without a gold-standard dataset. However, it remains unknown how the core architecture can be adjusted to incorporate graph convolution operations designed for GoW. Furthermore, embeddings generated from autoencoders on temporal data must be augmented with the diachronic computations to ensure appropriate comparisons of embeddings across semantic spaces. These methodological and domain gaps motivate the following research questions:

- What are the emerging exploit trends in online hacker forums?
- How do the semantics of hacker exploit terminology shift over time?
- What modifications do the core graph convolutions need to operate on Graph of Words (GoW)?

- How can the autoencoder be adapted to incorporate modified graph convolution operations?

5.4. Research Design and Testbed

We developed a novel CTI framework (Figure 30) to address the identified research gaps and answer the posited research questions. This framework has four major components: (1) Data Collection and Pre-Processing (2) Time-Spell Construction and Text Graph Representation, (3) D-GCAE Development, and (4) Exploit Trend Identification. Each component is summarized in the following subsections.

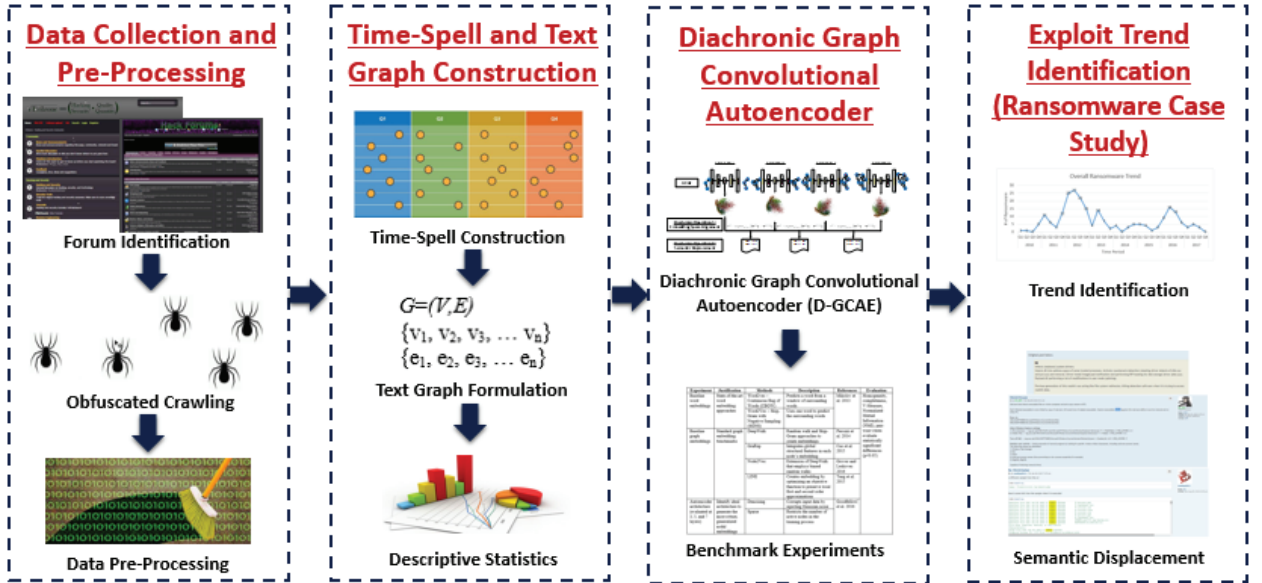


Figure 30. Research Framework for Identifying Emerging Hacker Exploit Trends

5.4.1. Data Collection and Pre-Processing

We collect two large and long-standing English hacker forums for analysis. These forums were selected based on suggestions by cybersecurity experts, their notoriety within the online hacker community for containing numerous malicious tools, and accessibility to all forum content without direct hacker invitations. Following forum identification, 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. A specialized Java program using Regular Expressions parses all post, thread, and author data into a relational database. Table 38 summarizes the final forum collection. All forum names are anonymized to protect ourselves from hackers within these communities.

Forum	Language	Date Range	# of Posts	# of Members	# of Exploits
O****C	English	2/07/2005 – 10/22/2017	137,664	7,011	2,720
K****e	English	3/11/2010 – 10/20/2017	24,933	1,418	7,245
Total:	-	2/07/2005- 10/22/2017	162,597	8,429	9,965

Table 38. Summary of Hacker Forum Collection

Our collection procedures resulted in a dataset with 162,597 posts (9,965 attached exploits) made by 8,429 hackers between 2005 and 2017. Exploit posts in the collection include ransomware, DoS, RATs, Zeus malware, buffer overflow, keyloggers, crypters, and others. Like other traditional social media platforms (e.g., Twitter, Reddit), hacker forum text has considerable inconsistencies and noise (Samtani et al. 2017). We address these issues with automated pre-processing procedures. First, all text is converted to lowercase to ensure different cases of the same word are not treated as distinct tokens. Second, all punctuation is stripped from the post content to remove extraneous, irrelevant characters. Third, a stop-words list filters all generic terms (e.g., the, at, this). Finally, Porter’s stemming algorithm unifies all remaining words to their root (e.g., “computer” becomes “comput”). These pre-processing steps are consistent with past literature employing word embedding approaches in hacker forum contexts (Benjamin and Chen 2015).

5.4.2. Time-Spell and Text Graph Construction

Conducting diachronic analysis requires the dataset in question to be split into multiple time-spells (Hamilton et al. 2016). Exploring the data in our collection indicates that 80-100 exploits are posted on a quarterly basis. As such, we split our dataset into time intervals of

three months (four quarters per year). This breakdown is also consistent with the analysis time-frames used by many industry CTI reports. To facilitate diachronic analysis, we construct a GoW in each time-spell. As discussed in the literature review, representing text in a GoW provides access to an array of metrics to understand overall network dynamics and pinpoint key network nodes. It also captures relationships between words missed in other text representation (e.g., vector space model) or word embedding approaches (e.g., word2vec). For these reasons, we adopt a GoW approach. Formally, each GoW is denoted as $G=(V,E)$. G is the entire undirected graph. V is the node set, $\{v_1, v_2, v_3, \dots v_n\}$ of all words appearing in exploit posts in that time-spell. E the edge set, $\{e_1, e_2, e_3, \dots e_n\}$. Nodes have an edge if they appear in the same forum post. The GoW in each time-spell builds on the previous period's. This formulation omits general, unrelated forum discussions while providing a granular look at exploit terms and their relationships.

5.4.3. *Diachronic Graph Convolutional Autoencoder (D-GCAE) Framework Development*

Constructing text graphs and computing network and node level metrics at each time-spell enables the identification of emerging trends, but cannot identify semantic shifts of terms (i.e., new meanings for existing words). Our review indicates GCNs are a state-of-the-art approach to generate node embeddings. However, their design is for supervised learning; thus, they generate task-specific embeddings. Autoencoders can generate task-independent embeddings, but cannot identify embedding shifts across time-spells. The limitations of existing GCN and autoencoder approaches, combined with existing diachronic computations, motivates the development of a novel deep learning approach to generate embeddings from GoW and map their shifts over multiple time-spells: the D-GCAE. The D-GCAE has two major components: GCAE and diachronic embeddings. The GCAE incorporates custom graph

convolution operations into the autoencoder to create task-independent (i.e., general) embeddings for GoW's. Figure 31 illustrates GCAEs' architecture and operations.

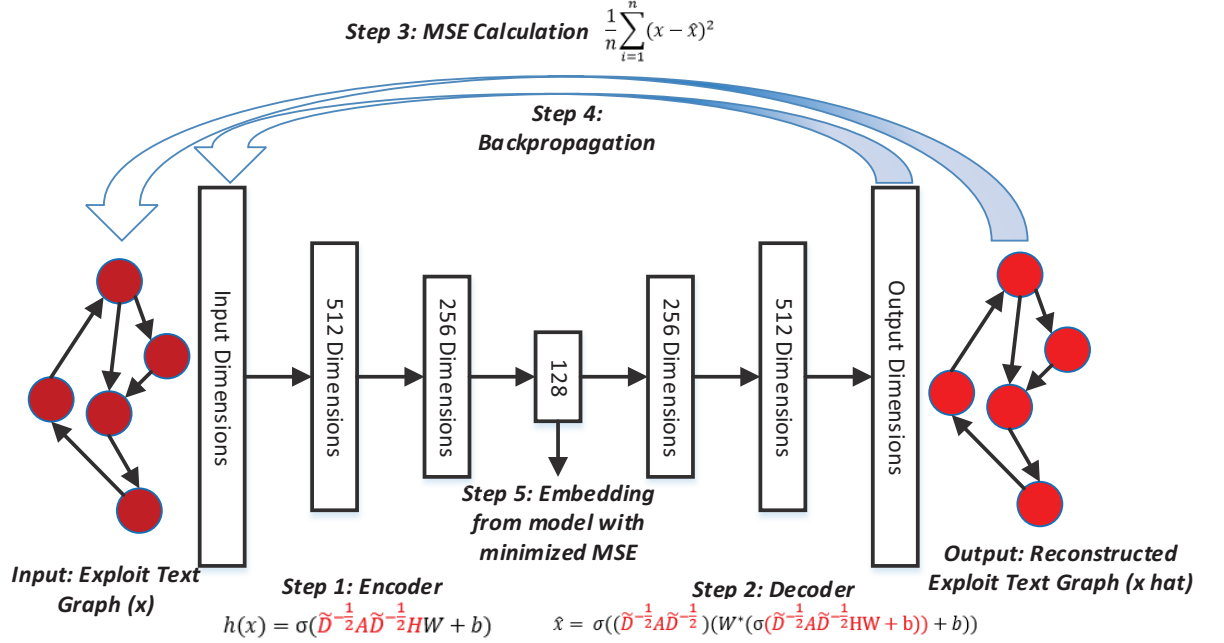


Figure 31. Architecture of the Graph Convolutional Autoencoder (GCAE)

From left: GCAE takes the GoW's adjacency matrix as input. Step 1 uses a modified graph convolution (denoted in red) and creates an embedding for each node. Step 2 reconstructs the original node by using a modified decoder function.

Step 3 compares the reconstructed output with the input using MSE. Step 4 backpropagates the error to adjust GCAE weights. This encoding, decoding, and backpropagation continues until MSE is minimized.

The GCAE component of the D-GCAE operates as follows. First, it receives the $N \times N$ adjacency matrix A from the GoW G . While GCNs can also account for nodal attributes part of the input, nodes in our formulation do not have any features. In accordance with Kipf and Welling (2017), an $N \times N$ identity matrix I is used in lieu of a feature matrix. After taking A and I as input, the GCAE creates a low-dimensional embedding for each node by extending the standard autoencoder's encoder computation, $h(x) = \sigma(Wx + b)$, to include graph convolutions. The extended encoder is:

$$h(x) = \sigma(\tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}} H W + b)$$

Where σ is the activation function, ReLU. H is the row-wise embedding of the graph nodes in a layer, W is the weight matrix, and b is the bias term. While Kipf and Welling (2017), used computed an alternate adjacency matrix, \tilde{A} , as part of the graph convolutions, we elect to use the standard adjacency matrix A . \tilde{A} was originally computed by adding an identify matrix I_N to capture self-loops and prevent any information loss. In the case of GoW, however, self-loops naturally occur; a word may appear in the same post twice. Adding an identify matrix would result in a redundancy of information that can skew the importance of words within the graph. In light of this, the identity matrix is removed. All other GCN operations remain the same. A and the degree matrix are multiplied to create a symmetric normalization, $\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}$ to prevent scaling issues. Each encoder layer uses these computations and reduces the previous layer's dimensionality. This continues until the condensed embedding is reached. The GCAE decoder then reconstructs the original input by extending the standard autoencoder decoder function of $\sigma(W^*(h(x)) + b)$ (where $h(x)$ is the encoder's output embedding) to:

$$\hat{x} = \sigma((\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}})(W^*(\sigma(\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}HW + b)) + b))$$

σ is the ReLU activation function, $\tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}$ is the symmetric normalization, W^* are the weight in matrix format, and b is the bias term. Each layer in the decoder applies these operations and also increases the dimensionality of the previous layer. After reaching the input dimensionality, MSE between the input x and reconstructed output \hat{x} is calculated and backpropagation adjusts GCAE weights as needed. Encoding, decoding, MSE calculation, and backpropagation continues until MSE is minimized. Once node embeddings are generated and tabulated into matrix form in each time-spell (i.e., $\mathbf{W}^{(t)} \in \mathbb{R}^{d \times |V|}$, where t is the time-spell). This final tabulation marks the completion of the GCAE component of the D-GCAE. Figure 32 visually illustrates the second part of the D-GCAE: diachronic word embeddings.

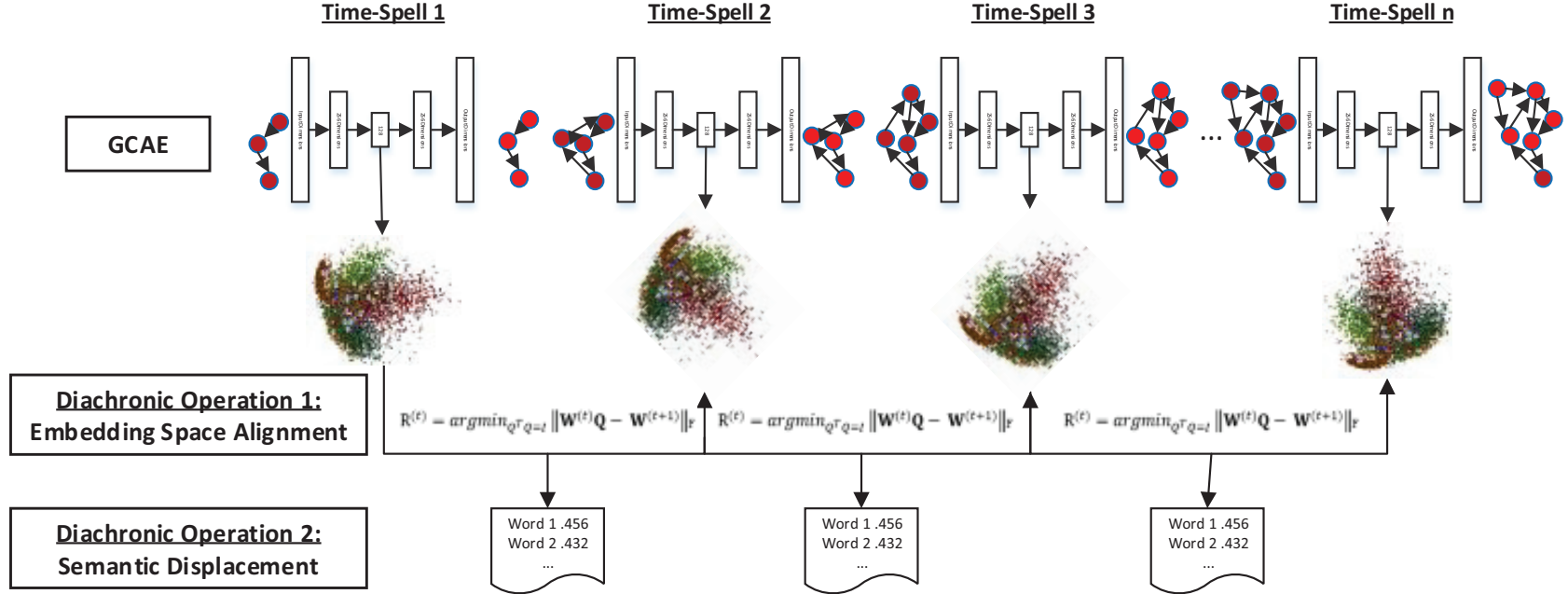


Figure 32. D-Graph Convolutional Autoencoder (D-GCAE) Procedure

The GCAE at each time-spell creates a low-dimensional embedding for each node. The first diachronic operation aligns all embedding spaces. The second operation calculates the semantic displacement of each word.

The diachronic aspect of the D-GCAE performs two tasks: align embedding spaces and compute semantic shifts. For the former, we adopt orthogonal Procrustes matrix operations as outlined by Hamilton et al. (2016). Specifically, embedding spaces across time-spells are aligned while retaining cosine similarities by optimizing the following objective function:

$$R^{(t)} = \underset{Q^T Q = I}{\operatorname{argmin}} \left\| \mathbf{W}^{(t)} \mathbf{Q} - \mathbf{W}^{(t+1)} \right\|_F$$

Where $\|\cdot\|_F$ denotes the Frobenius norm. Following this alignment, we compute the magnitude and rate of semantic displacements (i.e., a word’s semantic shift over time-periods) by calculating the cosine distance of a word’s embedding at across time-periods (i.e., $\cosine_dist(w_t, w_{t+\Delta})$). Results of these calculations help pinpoint emerging exploit terminology.

All D-GCAE computations described above, were implemented using the PyTorch, Numpy, Scikit-learn, Networkx, and Natural Language Toolkit (NLTK) packages in Python on a single Ubuntu Linux machine (4 GB RAM, 50 GB SSD). Algorithm 1 (below) summarizes all D-GCAE steps described above in a fashion that can be reproduced in other software and hardware environments.

ALGORITHM 1. Diachronic Graph Convolutional Autoencoder (D-GCAE)

Input: # of nodes N , adjacency matrix A ($N \times N$), # of encoder layers d_e , # of decoder layers d_d , embedding matrix for the previous time-spell $E' = [e'_1, e'_2, \dots, e'_M]$ ($M \leq N$), where e'_i is the embedding for node i

Output: embedding matrix for each node aligned to the previous embedding space $E^* = [e_1^*, e_2^*, \dots, e_N^*]$, where e_i is the embedding for node i

Procedure

$D = degree_matrix(A)$

$\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ //normalize the degree matrix

loop:

$H_1^{(e)} = X$

foreach GCAE encoder layer $(H_i^{(e)}, W_i^{(e)})$ ($i = 1, 2, \dots, d_e - 1$):

$H_{i+1}^{(e)} = ReLU(\hat{A} H_i^{(e)} W_i^{(e)})$ // graph convolution operations for encoder

$E = H_{d_e}^{(e)}$ // embedding generated by encoder


```

 $H_1^{(d)} = \text{ReLU}(\hat{A}H_{d_e}^{(e)}W_{d_e}^{(e)})$ 
foreach GCAE decoder layer  $(H_j^{(d)}, W_j^{(d)})$  ( $j = 2, 3, \dots, d_d - 1$ ):
     $H_{i+1}^{(d)} = \text{ReLU}(\hat{A}H_i^{(d)}W_i^{(d)})$  // graph convolution operations for decoder
 $\hat{X} = H_{d_d}^{(d)}$  // reconstructed output
 $err = \text{MSE}(X, \hat{X})$  // calculate error between reconstructed output and original input
backpropagate  $err$  to update  $W_i^{(e)}, W_j^{(d)}, (i = 1, 2, \dots, d_e; j = 1, 2, \dots, d_d)$ 
until  $err$  is minimized
 $E_M = E[e_1, e_2, \dots, e_M]$  //All embedding spaces
 $R = \text{argmin}_{Q|Q^T Q = I} \|R \cdot (E_M)^T - (E')^T\|_F$  // Frobenius norm
 $E^* = (R \cdot E^T)^T$ 
return  $E^*$ 

```

5.4.4. D-GCAE Evaluations

A critical aspect of computational design science research is the rigorous evaluation of a proposed artifact against clearly established benchmarks (Hevner et al. 2004). This essay's IT artifact is the D-GCAE framework. Fundamentally, the D-GCAE is an unsupervised approach to (1) generating word embeddings (GCAE) and (2) mapping their evolution over time (diachronic). Although possessing significant descriptive capabilities, evaluating diachronic operations is a major challenge. Since their advent by Hamilton et al. (2016), no direct alternatives or benchmarks have emerged. Direct evaluation of diachronic operations requires manually detecting semantic shifts in a corpus, then qualitatively identifying how well displacement calculations captured shifts (Hamilton et al. 2016; Yao et al. 2018). This ground truth knowledge is not available in most corpora. For these reasons, scholars have treated diachronic operations as an external mechanism to their model, and rather focused on evaluating the quality of the generated embeddings (in this study, the GCAE).

Recent word embedding literature has suggested two major evaluation approaches: extrinsic and intrinsic (Bakarov 2018). Extrinsic evaluations input word embeddings into a

selected downstream NLP task (e.g., text classification, clustering, POS tagging, etc.). Intrinsic evaluations, such as Precision-at-k ($P@k$), rely on human judgements of word embeddings. Between the two, extrinsic evaluations are often preferred, as using embeddings in a downstream task provides metrics (e.g., F-Measure, V-Measure) to quantitatively measure embedding quality for a salient analytics procedure.

The definition of an appropriate downstream task is often context specific. Within the CTI domain, professionals often wish to comprehensively understand an exploit's functions and implementation by examining closely associated terms (Bromiley 2016; Shackleford 2016). This knowledge can pinpoint new exploit names and emerging trends (e.g., Mirai for DDoS). These unique domain characteristics motivate the downstream task in which we evaluate the D-GCAE's word embeddings: k-means clustering. K-means is a popular clustering algorithm providing easily interpretable results in a computationally efficient manner (Hartigan & Wong, 1979). K-means' use of distance based metrics when creating groupings is ideal for 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 higher distance calculated between them, will be clustered apart, and the overall clustering performance suffers. This intuition has made k-means a popular approach to evaluating word embedding quality (Bakarov 2018; Zhai et al. 2016).

Using k-means as a downstream evaluation task requires a gold-standard dataset. As such, we created three gold-standard clusters of exploit posts (30 each) from one forum in our collection: keyloggers, crypters, and RATs. These categories were selected as they have

minimal overlap in terms of technical functionality and implementation. Adhering to best practices, we asked a panel of four cybersecurity students and one professor to validate the exploits in each category. If a panelist disagreed with our interpretation, we asked him/her to provide an alternate suggestion. 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 Cronbach’s alpha statistic, reaching a final value of 0.9846. Given the high level of concordance between raters, we construct a text graph on the selected postings. The gold-standard network serves as a testbed for three experiments that evaluate the performance of the GCAE and its components against baseline embedding approaches. All three are summarized in Table 39.

Experiment	Justification	Methods	Description	References	Evaluation
Baseline word embeddings	State-of-the-art word embedding approaches	Word2vec – Continuous Bag of Words (CBOW)	Predicts a word from a window of surrounding words	Mikolov et al. 2013	Homogeneity, completeness, V-Measure, Normalized Mutual Information (NMI); paired t-tests
		Word2Vec – Skip-Gram with Negative Sampling (SGNS)	Uses one word to predict the surrounding words		
Baseline graph embeddings	Standard graph embedding benchmarks	DeepWalk	Random walk and Skip-Gram approaches to create embeddings	Perozzi et al. 2014	evaluate statistically significant differences (p<0.05)
		GraRep	Integrates global structural features in each node’s embedding	Cao et al. 2015	
		Node2Vec	Extension of DeepWalk that employs biased random walks	Grover and Leskovec 2016	
		LINE	Creates embedding by optimizing an	Tang et al. 2015	

			objective function to preserve local first and second order approximations		
Autoencoder architecture (evaluated at 3, 5, and 7 layers)	Identify ideal architecture to generate the most robust, generalized nodal embeddings	Denoising	Corrupts input data by injecting Gaussian noise	Goodfellow et al. 2016	
		Sparse	Restricts the number of active nodes in the training process		

Table 39. Summary of Benchmark GCAE Experiments

Experiment 1 compares GCAE against the prevailing word embedding approach not employing text graphs: word2vec. Introduced by Mikolov et al. (2013), word2vec trains shallow, two layer neural network to reconstruct linguistic contexts of words. Word2vec takes as input a corpus of data, and generates an embedding for each word. Word2vec can use two model architectures: Continuous Bag-of-Words (CBOW) and Skip-Gram with Negative Sampling (SGNS). CBOW predicts a word based on a window of surrounding words. Conversely, SGNS uses one word to predict surrounding words. Generally speaking, CBOW is faster, but SGNS creates better word embeddings for infrequent words.

Experiment 2 benchmarks the GCAE embeddings against four node embedding approaches: Large-scale Information Network (LINE), Graph Representations (GraRep), DeepWalk, and Node2Vec. LINE captures first and second order approximations of each node while preserving local and global graph structures to optimize an objective function (Tang et al. 2015). GraRep uses global network features when creating each node's embedding (Cao et al. 2015). DeepWalk mimics the Skip-Gram approaches used in word2vec along with random graph walks to learn node embeddings (Perozzi et al. 2014). Node2vec extends DeepWalk by

using a biased random walk rather than a pure one to create embeddings (Grover and Leskovec 2016).

Experiment 3 discovers how modifications to the core GCAE affect embedding quality. Deep learning literature posits that novel architectures should be evaluated against alternate designs in terms of variations to the core structure and depth of the model (Goodfellow et al. 2016). By default, the GCAE uses a fully-connected autoencoder. Experiment 3 evaluates the default GCAE against two alternatives: denoising and sparse. Denoising architectures inject Gaussian noise into the input data. In theory, this forces the autoencoder to learn a more robust embedding of the input data. Sparse autoencoders restricts the number of active nodes during training. Past studies have shown that imposing denoising and sparsity constraints create richer embeddings. However, it is unclear whether these benefits are seen in an autoencoder employing custom graph convolutions on a GoW. Adhering to best practices, the performances each D-GCAE's is evaluated at three, five, and seven layers.

Four well-established metrics evaluate the quality of the clusters in each experiment: completeness, homogeneity, V-Measure, and Normalized Mutual Information (NMI). 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 (Rosenberg and Hirschberg 2007). NMI measures cluster quality by identifying the number of word embeddings have the same labels (i.e., mutually shared information) between clusters. All four metrics calculate a scalar value between 0.0 (random assignment) and 1.0 (perfect match). In all experiments, paired t-tests identify statistically significant differences between methods. Differences were considered statistically significant if $p < 0.05$.

5.4.5. Exploit Trend Identification: Case Study

The last component of our framework illustrates D-GCAE's practical utility with an in-depth case study of one English forum. The steps used to execute this case study simulates the process a CTI professional can take when aiming to identify exploit trends and terms. First, the data is split into equal time-spells of three months. A GoW is created within each period. Nodes represent all words in exploit posts in that time-spell, and edges denote if two words appeared in the same post. Network level statistics (e.g., network diameter, clustering coefficient, etc.) and node level centralities are calculated for each graph. The D-GCAE is then applied to first create embeddings in each time-spell, then calculate each word's semantic displacement (i.e., how much words shift in their meaning) to identify emerging exploit trends.

5.5. Results and Discussion

5.5.1. Experiment 1: GCAE vs Word2vec

Experiment 1 evaluated the GCAE operations against two variants of word2vec: CBOW and SGNS. Both word2vec implementations used the default hyperparameter settings as provided by the Gensim Python package. Table 40 and Figure 33 summarize full evaluation results.

Embedding Approach	Homogeneity	Completeness	V-Measure	NMI
3 layer GCAE	44.935%	54.147%	49.113%	48.355%
Word2vec (SGNS)	30.747%*	31.257%*	31.000%*	30.969%*
Word2vec (CBOW)	30.191%*	30.189%*	30.190%*	30.190%*

Table 40. GCAE vs Word2Vec Evaluation Results
(* indicates statistically significant difference at $p < 0.05$)

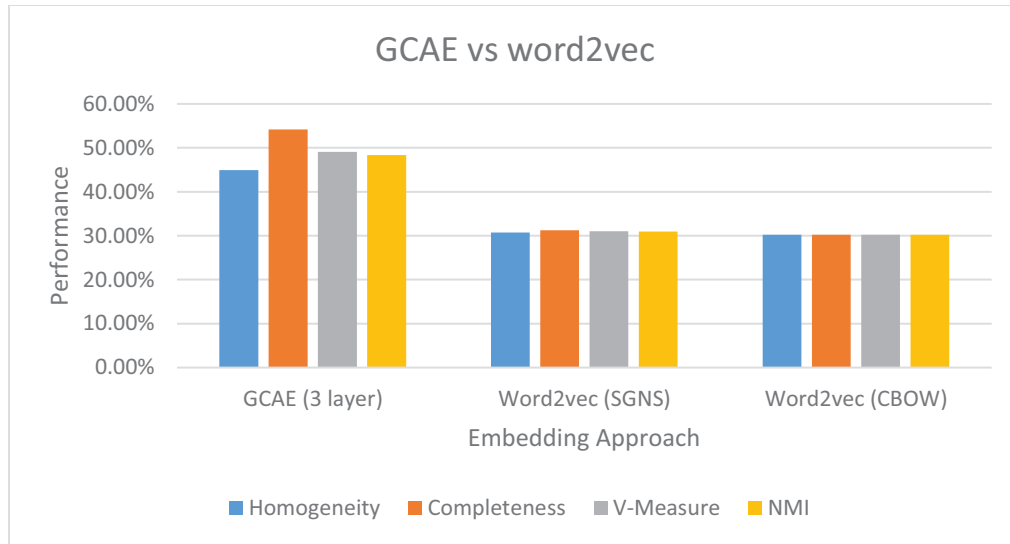


Figure 33. GCAE vs Word2Vec Evaluation Results

Overall, the GCAE outperformed both word2vec variants in terms of homogeneity, completeness, V-Measure, and NMI by statistically significant margins ($p < 0.05$). SGNS and CBOW achieved similar V-Measures (harmonic mean of homogeneity and completeness) at 31.000% and 30.190%, respectively. These performances suggests GCAE's usage of a GoW captures relationships missed by word2vec. Examining the text representation each algorithm uses can provide explanations. Word2vec draws its foundations from the sub-discipline of distributional semantics. This paradigm assumes a word's representation can be created by examining its surrounding words. Word2vec uses a window approach, where a word's embedding is created by predicting surrounding words (CBOW) or using surrounding words to predict a specific term (SGNS). This approach suffers on shorter posts, where the appropriate context is not available. This issue compounds if the word appears infrequently in the corpus. GoW, however, can still capture a word's relationships to others within and across forum posts (irrespective of its frequency). Future work can compare the GCAE against word2vec on larger, traditional text corpora (e.g., Google Books) to gain further insight into this phenomena.

5.5.2. Experiment 2: GCAE vs Text Graph Embeddings

Experiment 2 evaluated the GCAE with custom graph convolutions against state-of-the-art unsupervised graph approaches: DeepWalk, Node2Vec, LINE, and GraRep. All baseline approaches used default hyperparameter setting provided by the OpenNE Python packages. To ensure consistency of comparisons, all methods generated 128 dimension embeddings. Full evaluation results are presented in Table 41 and visualized in Figure 34.

Embedding Approach	Homogeneity	Completeness	V-Measure	NMI
GCAE (3 layer)	44.935%	54.147%	49.113%	48.355%
LINE	37.770%*	37.741%*	37.756%*	37.722%*
Node2Vec	36.418%*	31.248%*	33.635%	38.949%*
DeepWalk	24.664%*	29.782%*	26.982%*	26.754%*
GraRep	23.952%*	26.541%*	25.180%*	25.084%*

Table 41. GCAE Benchmark Evaluation Results
(* indicates statistically significant difference at $p < 0.05$)

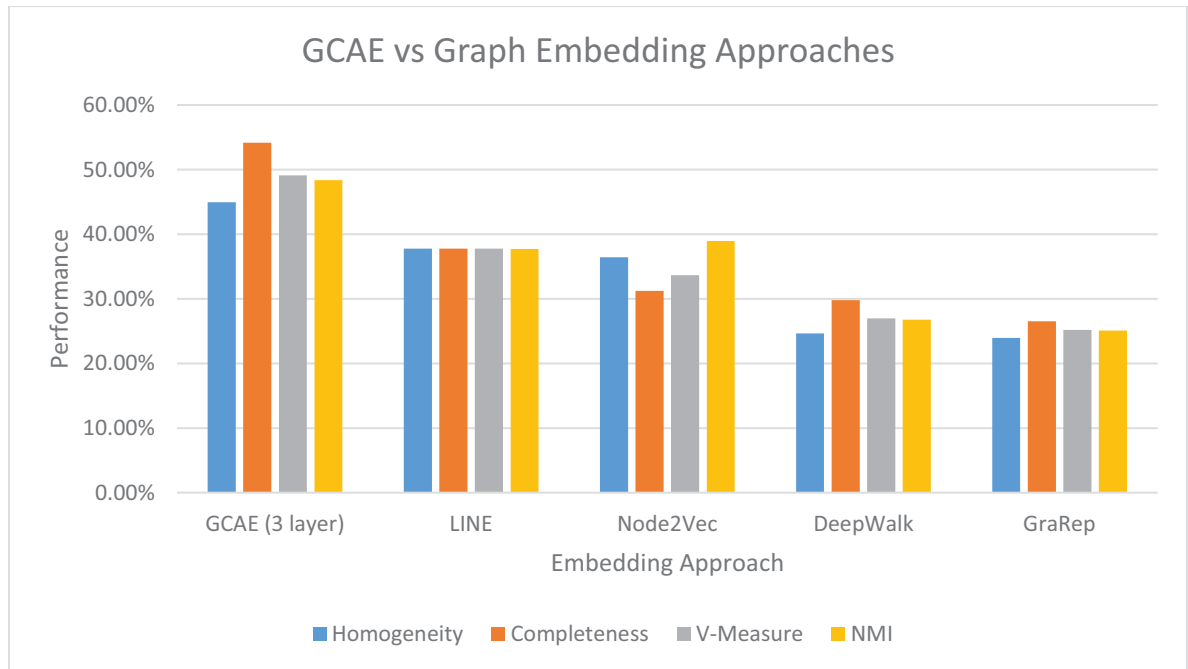


Figure 34. GCAE Benchmark Evaluation Results

Overall, the GCAE with the custom graph convolution outperformed LINE DeepWalk, Node2Vec and GraRep in terms of homogeneity, completeness, V-Measure, and NMI. All margins were statistically significant. GCAE's performance may be explained by how it

operates compared to baseline approaches. LINE optimizes an objective function using first and second order node approximations, DeepWalk and node2vec use skipgram approaches with random or biased walks (respectively), and GraRep only uses global features. Each neglects to scan the network either at a global or local level. Those methods scanning locally (node2vec, DeepWalk, and LINE) often employ sampling and approximation approaches, rather than looking at each node individually. Such approaches scale well to larger networks, but may sacrifice performance. In contrast, GCAE examines each node and its neighborhood via convolution operations. GCAE also incorporates overall network features (e.g., degree matrix) when generating embeddings. Taken together, these procedures capture more information about each node.

5.5.3. Experiment 3: GCAE vs Sparse and Denoising GCAE

Experiment 3 evaluated D-GCAE’s fully connected architecture against sparse and denoising alternatives at varying levels of depth. To ensure consistency of comparisons, each GCAE architecture was trained for 200 epochs (i.e., 200 passes over the data), imposed a dropout of 0.5, used a learning rate of 0.01, and employed the ReLU activation function. Each approach generated 128 dimension embeddings. Full results are summarized in Table 42 and visualized in Figure 35.

GCAE Configuration	GCAE Depth	Homogeneity	Completeness	V-Measure	NMI
GCAE	3 layer	44.935%	54.147%	49.113%	48.355%
	5 layer	41.741%	43.877%*	42.782%*	42.598%*
	7 layer	51.225%	51.423%	51.324%	51.321%
Sparse GCAE	3 layer	30.179%*	43.413%*	35.494%*	31.548%
	5 layer	30.020%*	44.849%*	35.966%*	31.714%*
	7 layer	2.558%*	5.115%*	3.410%*	2.776%*
Denoising GCAE	3 layer	44.564%	52.617%	48.256%	47.588%
	5 layer	45.226%	57.053%	50.456%	49.441%
	7 layer	43.964%	52.408%	47.816%	47.013%

Table 42. GCAE Architecture Evaluation Results

(* indicates statistically significant difference at $p < 0.05$ compared to the 3 layer GCAE)

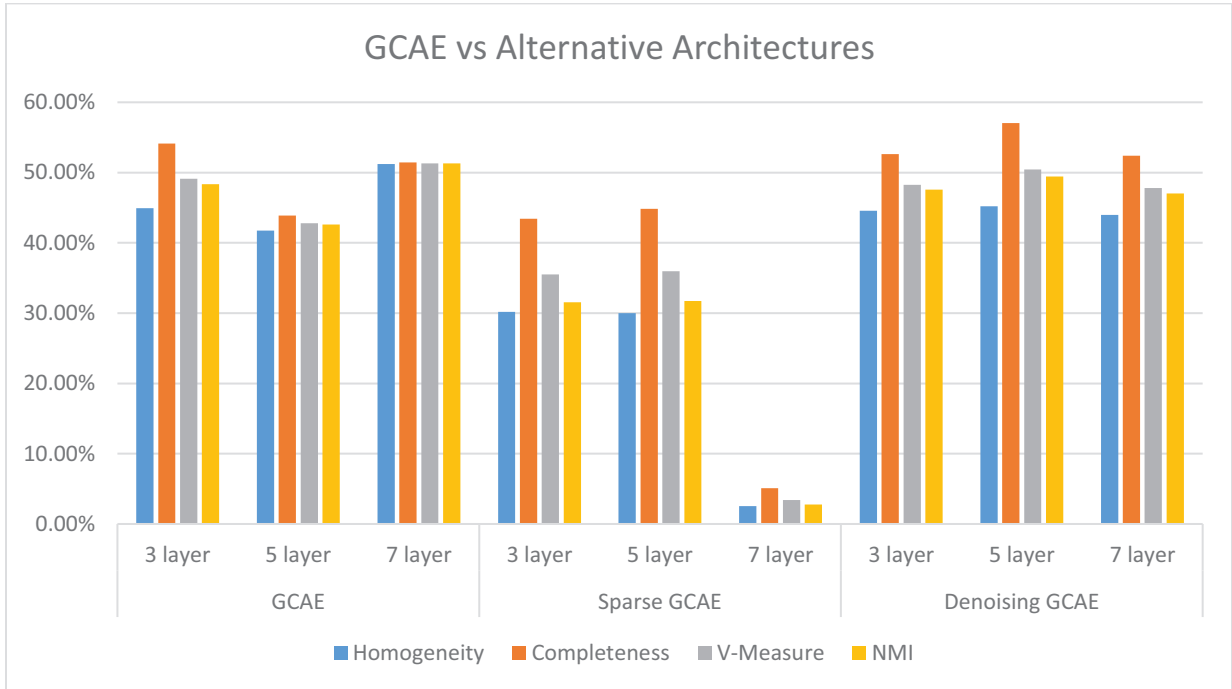


Figure 35. GCAE Architecture Evaluation Results

Overall, the standard 7 layer GCAE outperformed other configurations in terms of homogeneity (51.225%), V-measure (51.324%), and NMI (51.321%). The five layer denoising GCAE outperformed all variations in terms of completeness (57.053%). These results reveal two key insights that can guide future researchers in their D-GCAE architecture selection. First, the sparse GCAE achieved significantly lower performances than the standard and denoising architectures. The graph convolutions used by the GCAE provides a possible explanation. Unlike the standard autoencoder that uses feed-forward computations, GCAE's incorporate additional computations (e.g., degree matrix integration). Restricting the activation of neurons affects the GCAE's ability to effectively learn each node's embedding.

The second insight from experiment 3 is the relationship between depth and performance. Across the board, each GCAE's performance improved as depth increased from three layers to five. This trend continued for the standard GCAE when the depth increased to

seven. However, this did not hold true for the sparse and denosing variations, which saw lower performances with seven layer architectures. These results suggest that using deeper GCAE's can aid overall performance, but at a certain threshold, additional depth may deteriorate overall performance. It is likely that encoder layers is deeper GCAE's with lower performances remove the most important features such that the decoder reconstructs the input using an over-summarized embedding.

5.5.4. Ransomware Case Study Results

The forum selected for our case study contains 24,933 posts (7,245 with attached exploits) made by 1,418 hackers between 3/11/2010 and 10/20/2017. 235/7,245 (3.2%) are ransomware. Two reasons can explain this low percentage. First, ransomware requires in-depth systems knowledge (e.g., memory processes, file directory structures/paths). Casual forum members (e.g., script kiddies) often do not possess this expertise. Second, compared with other well-established exploits (e.g., bots), ransomware remains a relatively new technology, emerging as a notable threat in the early 2010's. Table 43 summarizes the number of ransomware exploits detected in each quarter of the forum's existence (2010 – 2017). Figure 36 provides a visual representation of the trends.

Quarter	Date Range	2010	2011	2012	2013	2014	2015	2016	2017	Total
		Q1-Q4	Q5-Q8	Q9-Q12	Q13-Q16	Q17-Q20	Q21-Q24	Q25-Q28	Q29-Q32	Q1-Q32
1	1/1-3/31	1	11	25	4	4	5	9	3	62
2	4/1-6/30	1	6	27	14	0	4	16	5	73
3	7/1-9/30	0	3	22	7	3	1	13	3	52
4	10/1-12/31	5	12	15	2	5	3	6	-	48
Total:	01/01 – 12/31	7	32	89	27	12	13	44	9	235

Table 43. Number of Ransomware Per Quarter (2010 – 2017)

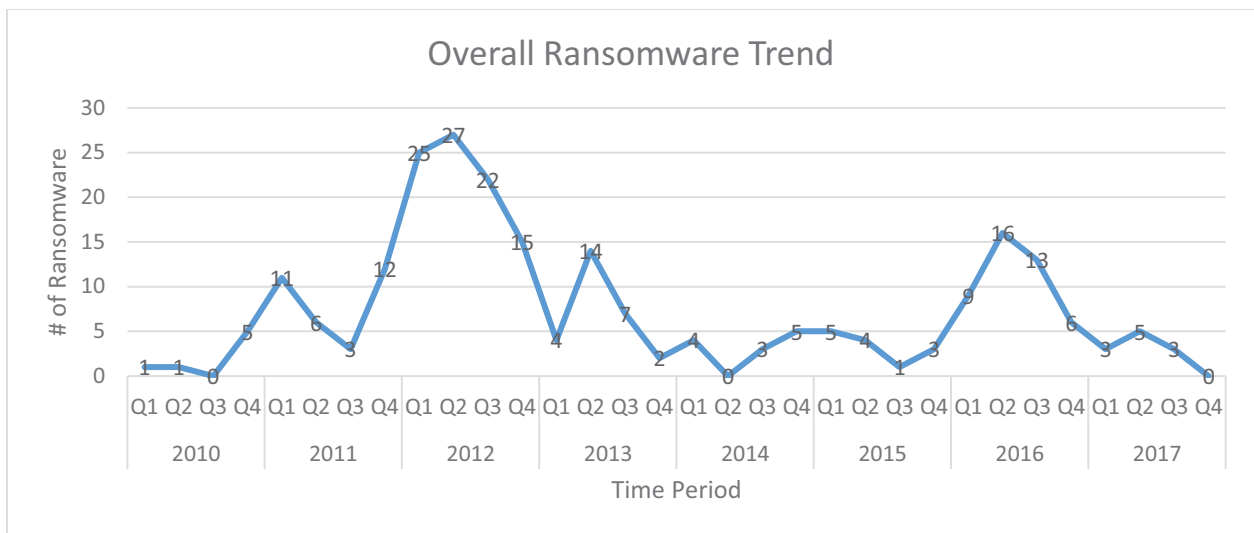


Figure 36. Overall Ransomware Trend (2010 – 2017)

Ransomware rose to prominence in the selected forum in quarters 10-12 of 2012. While many explanations can describe the spike, this surge coincides with the creation of the seminal Reveton and Cryptolocker. Each appears in our collection. These variants focused solely on encrypting a victim’s computer and demanding ransom. Ransomware postings tapered off in 2014-2015, before enjoying a renaissance in quarters 2 and 3 in 2016. Ransomware at this point evolved to incorporate more robust functionalities such as backdoors and mobile device (e.g., Android) capabilities. 2016 also marked the first appearance of Ransomware-as-a-Service (RaaS), where unskilled hackers could launch attacks without needing in-depth technical knowledge.

Plotting ransomware trends based on has the same limitations as prior studies: inability to identify the overall ransomware landscape and detect relationships between ransomware strains. These issues are countered by creating text graphs at each time-slice. Each graph builds upon the previous. We calculate node and network level measures to understand overall network dynamics. Results are summarized in Table 44. For space considerations, we only list year-end results.

Category	Metric	2010	2011	2012	2013	2014	2015	2016	2017
Forum	# of ransomware	7	39	128	155	167	180	224	235
Network Level Metrics	# of nodes	334	569	935	1,040	1,109	1,225	1,463	1,511
	# of edges	27,742	36,918	48,017	50,663	53,070	60,058	69,129	70,117
	Network diameter	2	3	3	4	4	4	4	4
	Radius	2	2	2	2	2	2	2	2
	Graph density	0.499	0.228	0.110	0.094	0.086	0.080	0.065	0.061
	Avg. path length	1.501	1.818	1.980	2.008	2.027	2.037	2.060	2.076
	Avg. clustering coefficient	0.967	0.922	0.888	0.882	0.881	0.879	0.877	0.878
Node Level Metrics	Minimum degree	10	4	2	2	2	2	2	1
	Maximum degree	220	397	606	681	700	804	1,005	1,038
	Average degree	166.12	129.764	102.710	97.429	95.708	98.054	94.503	92.809

Table 44. Topological and Node Level Descriptive Statistics Between 2010 – 2017

Note: The numbers in each time-spell accounts for the total in previous spell(s)

Table 44 reveals several key insights not available in Figure 36. The surges seen in 2012 and 2016 were met with an increase in the number of nodes (i.e., words) (569 to 935 from 2011 – 2012 and 1,109 to 1,225 from 2015 – 2016). This growth indicates that hackers are using a richer ransomware lexicon. The increase in vocabulary size was concurrent to the decrease in overall graph density (0.499 to 0.065) and average clustering coefficient (0.967 to 0.877). The decrease in both measures indicates that hackers diversify their ransomware interests and specialties, resulting in new ransomware implementing novel functionalities. The increases in average path length and network diameter support this observation.

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. To gain insight into the key words into the network (which can point to core features used by all ransomware), we summarize the top ten words based on degree and betweenness centralities in Table 45. For space considerations, we only list the ranks of words at the year-ends of 2015 – 2017. We note that the words listed Table 45 are in their stemmed format.

Degree Rank	2015				2016				2017			
	Word	Degree	B. Rank	Between.	Word	Degree	B. Rank	Between.	Word	Degree	B. Rank	Between.
1	ransomwar	804	2	75,801.94	ransomwar	1,005	1	149,974.14	ransomwar	1,038	1	172,243.98
2	name	689	5	28,869.30	ransom	807	2	126,390.70	ransom	807	2	128,709.01
3	ransom	681	1	92,723.22	malwar	713	4	40,868.70	malwar	731	4	47,056.43
4	malwar	661	4	33,305.32	name	712	6	31,762.20	name	712	6	32,033.92
5	window	619	8	15,017.44	file	685	5	34,628.16	file	685	5	34,711.40
6	insid	562	12	10,911.77	window	619	13	14,054.68	window	619	15	14,169.65
7	sampl	527	3	38,335.50	system	584	10	14,795.97	system	584	13	14,910.02
8	file	520	7	16,554.16	insid	570	18	11,730.08	insid	570	19	11,901.05
9	detect	518	15	9,925.35	sampl	563	3	46,578.41	download	565	9	47,262.15
10	dropper	499	25	5,619.74	dropper	553	31	7,701.41	sampl	563	3	7,094.09

Table 45. Topological and Node Level Descriptive Statistics Between 2014 – 2017

The top ten words in terms of degree centrality, such as “ransomwar,” “name,” “malwar,” “detect,” and “file,” remain relatively unchanged at each year-end. This indicates that even if new strains emerge ransomware’s core functionalities often remains the same (e.g., deploying malware, demanding ransom). CTI professionals can use this knowledge to discern ransomware’s capabilities and operations (knowledge often gleaned by malware analysis). While useful, the intelligence provided by Table 45 does not reveal the specific features incorporated over the years to develop new ransomware. We gain deeper insight into this by employing diachronic semantic displacement calculations to compute the average amount a word shifts between time-spells. The top 20 words with the highest average shift are summarized in Table 46. As with Table 45, all words in Table 46 are presented in their stemmed format.

Rank	Word	Amount Shifted*	Rank	Word	Amount Shifted*
1	Initiial	1.56378	11	Instal	1.53015
2	Variant	1.55563	12	Host	1.52990
3	Steal	1.55430	13	Vari	1.52902
4	Touch	1.55418	14	Strategi	1.52778
5	Organ	1.54652	15	Case	1.52593
6	Summer	1.53727	16	Financi	1.52273
7	Wolf	1.53707	17	August	1.52201
8	Mine	1.53594	18	Major	1.51968
9	Bitcoin	1.53217	19	Establish	1.51908
10	multicompon	1.53145	20	infect	1.51857

Table 46. Top Shifted Words Between 2010 – 2017
 (* average shift per time-spell)

Each word in the corpus shifted an average amount of 1.2538 in semantic space. Words with the top 20 average shifts relate to specific ransomware functionalities (e.g., “bitcoin” for payment mechanisms, “steal,” “variant,” “organ” for exploitation). One top word providing actionable intelligence is “infect,” which appears at rank 20 (shift of 1.51857). “Infect” appeared in various forms (e.g., “infect,” “infected,” “infection,” etc.) in a total of 19/235

(8.0551%) ransomware postings. Figure 37 provides a representative sample post from three time points, 2010, 2014, and 2017, to illustrate how infect’s meaning has shifted.

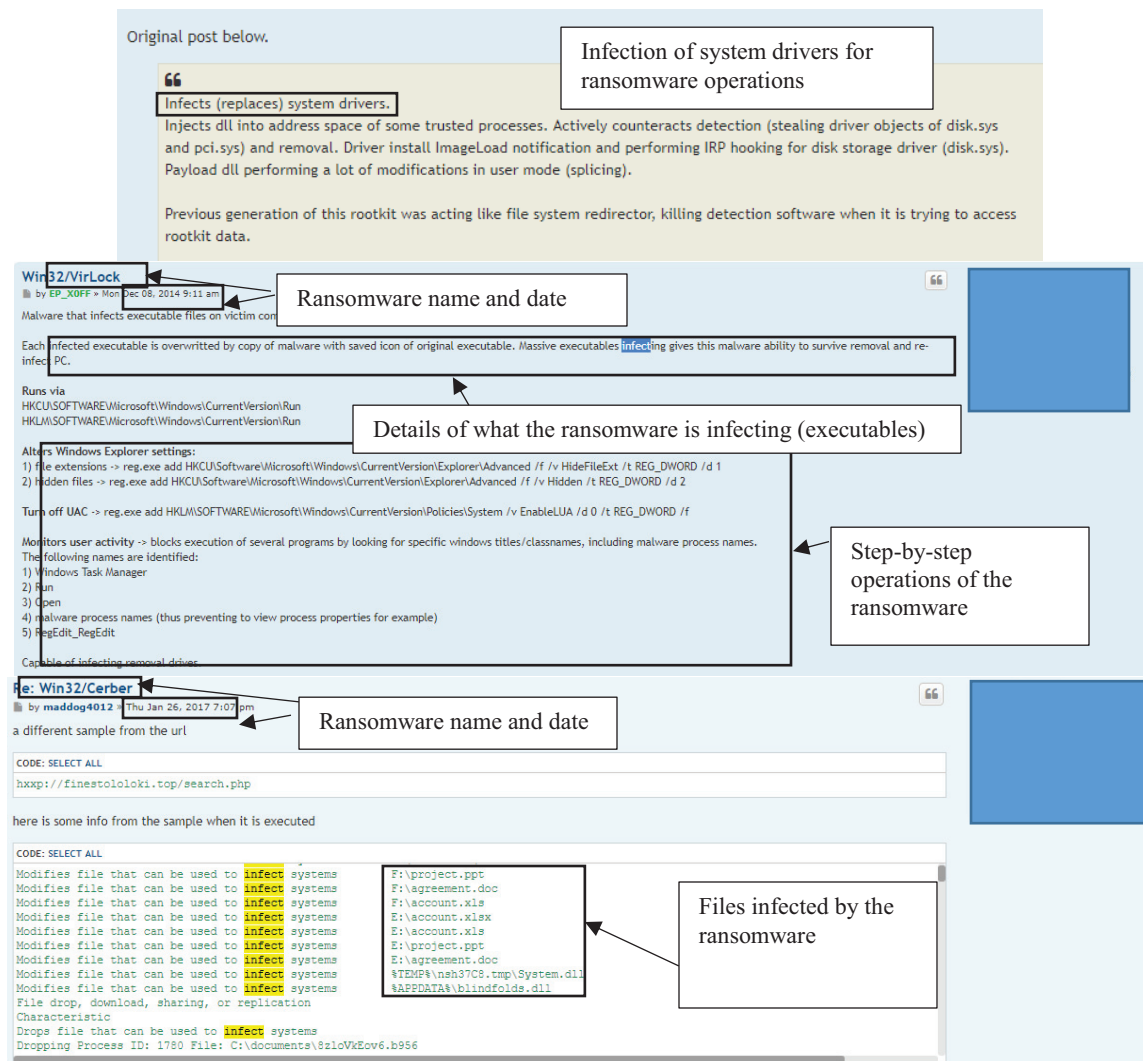


Figure 37. Three representative sample ransomware posts 2010 (top), 2014 (middle), and 2017 (bottom) to illustrate how the term “infect” has changed in meaning. Top ransomware focused specifically on memory injections. Middle adds in capabilities to infect executables on the victim machines. The bottom post focuses on infecting specific files.

The top post indicates that the term infect pertained to system drivers, specifically focusing on injecting Dynamic Link Libraries (DLL’s) into the memory address space to steal driver objects. The VirLock ransomware in the middle post moves beyond infecting memory processes to causing damage to executables on a victim’s machine. This enables attackers to

alter Windows Explorer settings and monitor user activities prior to encryption. Such capabilities can facilitate Advanced Persistent Threat (APT's), or attacks where an adversary remains undetected in an organization's network for extended periods of time. The final post provides the Cerber ransomware. This strain creates and/or modifies specific files (e.g., "project.ppt," "account.xlsx") to infect the victim's computer.

Existing methods of identifying emerging threats presented cannot reveal these shifts. However, identifying them can provide valuable tactical leads for CTI professionals (such as those monitoring hospitals, an industry widely afflicted with ransomware). 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). SIEM's are used by many CTI professionals to monitor the status of machines on a network and detect Indicators of Compromise (IoC's). Using the intelligence provided by third post, the SIEM would monitor if the listed file names appear on devices within their network. Should they exist, the SIEM can quarantine the machine and generate alerts to systems administrators who can mitigate the threat.

5.6. Conclusion 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 exploits already used in cyber-attacks. Consequently, cyber-attacks 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 exploits. Despite its promise, hacker forum's unique data characteristics necessitates novel, customized CTI analytics.

In this essay, we examine online hacker forums to identify emerging exploit trends. To do this, we developed the D-GCAE (Diachronic Graph Convolutional Autoencoder), a novel deep learning architecture that incorporates modified graph convolutions from the emerging GCN method into the autoencoder. This novel architecture operates on a GoW (Graph-of-Words) representation of hacker forum exploit text to create low-dimensional word embeddings in an unsupervised fashion. Semantic displacement measures adopted from diachronic linguistics literature map the evolution of hacker terminology over multiple time-spells. A series of benchmark evaluations reveals D-GCAE’s superior performance over prevailing 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 task.

D-GCAE’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 ransomware was posted. More importantly, we pinpointed how specific words shifted in their meaning. Identifying these semantic displacements helped detect new ransomware functionalities. Each discovery can provide valuable, actionable CTI for selected professionals to proactively create more robust cyber-defenses. While demonstrated in hacker forums, the D-GCAE can be leveraged by scholars for other novel, high-impact network science applications.

There are several promising directions for future work. First, scholars can fuse 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 exploits propagate through cyberspace. Second, future work can employ predictive analytics to forecast exploit

trends at future time-spells. Third, law enforcement can identify key hackers behind the emerging exploits and pursue investigations accordingly. Finally, organizations can create linkages between discovered exploits and their vulnerabilities to develop comprehensive cyber-defenses. Each direction can develop proactive CTI capabilities to ultimately create a safer and more secure society.

6. CONCLUSION AND FUTURE DIRECTIONS

The regular and disturbing frequency of cyber-attacks has made cybersecurity a grand societal challenge. Cyber Threat Intelligence (CTI), or the systematic process of identifying emerging threats and key threat actors to enable effective cybersecurity decisions, has become a viable approach to combat this ever increasing threat. Despite CTI's promise and potential value, many respected CTI professionals from cybersecurity firms such as the SANS Institute and other major industry firms (e.g., EY, KPMG) have noted that existing CTI procedures are too reactive and high-level in nature. To combat these issues, experts have suggested (1) examining data generated from the online hacker community to learn directly from the hackers and (2) employ machine learning processes to gain deeper insight into collected data. While IS scholars are well-positioned make significant advances in both areas, there remains a current dearth of work addressing these gaps.

This dissertation takes an important first step in the IS discipline in creating proactive CTI capabilities. Through the course of four essays, this dissertation systematically explored exploits in online hacker forums by developing and demonstrating a series of novel computational IT artifacts. Each artifact (and the related research results) offers practical value to CTI professionals and law enforcement while simultaneously providing value, guidance, and contribution to the IS knowledge base to support future researchers in their selected endeavors. Both types of contributions are summarized below.

6.1. Contributions: Practical and to IS Knowledge Base

Numerous IS scholars throughout the discipline's history have posited that IT artifacts generated from design science research should: (1) contribute salient descriptive knowledge and value to the business environment from which the problem was drawn and (2) provide

prescriptive knowledge to the IS knowledge base such that future scholars can pursue novel, high-impact research (Nunamaker et al. 1990; Peffers et al. 2007; Hevner et al. 2004; Gregor and Hevner 2013; Rai 2017). Each essay's IT artifact to addressed a salient CTI issue while simultaneously contributing design principles, instantiations, and nascent design theories to the IS knowledge base. Table 47 summarizes each essay's core research question, the IT artifact designed to address the question, the practical value provided by the artifact, and the contributions the artifact makes to the IS knowledge base for future researchers. Each is then discussed in further detail.

Essay	Research Question	IT Artifact(s)	Practical Contributions	Contributions to IS Knowledge Base
I	What exploits exist within hacker forums?	AZSecure Hacker Assets Portal	<ul style="list-style-type: none"> - System for CTI analysts to explore hacker exploits 	<ul style="list-style-type: none"> - Text analytics procedures to identify and categorize hacker forum exploits
II	What vulnerabilities do hacker forum exploits target?	Exploit Vulnerability DSSM; Device Vulnerability Severity Metric (DVSM)	<ul style="list-style-type: none"> - Enable device prioritization - Enable efficient device investigations 	<ul style="list-style-type: none"> - Richer semantic representations to enable accurate and relevant linkages - Capturing the decay in value of an exploit via mathematical representations - Ideal word hashing representation of cybersecurity text.
III	Who are the key hackers and exploit sharing communities in forums?	Graph Convolutional Autoencoder (GCAE)	<ul style="list-style-type: none"> - Identify groups of exploit sharing hackers - Pinpoint current key exploit sharing hackers for investigations 	<ul style="list-style-type: none"> - Shallower GCN architectures outperform deeper ones. - Sparse and denoising architectures harm GCAE embedding quality - Structural and nodal attributes can be captured within the same 128 dimension
IV	What are the emerging hacker exploits?	Diachronic - Graph Convolutional Autoencoder (D-GCAE)	<ul style="list-style-type: none"> - Detect emerging threats - Identify new exploit functionalities 	<ul style="list-style-type: none"> - Novel approach for generating word embeddings - Ability to map the evolution of language over time based on graph representations

Table 47. Each Dissertation Essay's Contributions to Practice and the IS Knowledge Base

The first essay sets the foundation for the remainder of the dissertation by asking the question “what exploits exist within hacker forums?” This essay presented a principled web, data, and text mining framework for analyzing (identifying and categorizing) hacker forum exploit content. This framework serves as a foundation for a novel CTI platform (i.e., this essay’s IT artifact), the AZSecure Hacker Assets Portal. This system provides valuable intelligence capabilities to CTI professionals both within public and private sectors to glean in-depth insight from the online hacker community without directly accessing the platforms from which the exploits originate. The framework powering the Hacker Assets Portal provides prescriptive knowledge to IS scholars wishing to examine forums by highlighting how text analytics procedures can be customized specifically for extracting intelligence hacker community data.

Receiving feedback from numerous end-user groups such as the National Cyber Forensics and Training Alliance (NCFTA) and Policing in Cyberspace (POLCYB) on essay I’s output highlighted a glaring issue: simply identifying what hacker exploits exist has minimal CTI value. An organization must be aware of which hacker exploits relates to their vulnerabilities. This feedback motivated essay II’s core research question: “what vulnerabilities to hacker exploits target?” This essay’s IT artifact, the Exploit-Vulnerability Deep Structured Semantic Model (EV-DSSM) was designed to address this question. The EV-DSSM leverages key text characteristics provided by hacker forum exploit text (target of exploit) and vulnerability assessment descriptions to identify the most relevant hacker exploits for known vulnerabilities detected by modern vulnerability assessment tools (e.g., Nessus). Beyond the contribution of the EV-DSSM algorithm, this essay also presented the Device Vulnerability Severity Metric (DVSM). The DVSM incorporates key cybersecurity features

(e.g., exploit age, vulnerability severity) to enable cybersecurity professionals operating the EV-DSSM to prioritize devices on their network. Both the EV-DSSM and DVSM's practical utility were demonstrated with in-depth case studies of hospitals and Supervisory Control and Data Acquisition (SCADA) systems. The design principles of presented in this essay, specifically those of extracting salient information to supplement exploit vulnerability linkages and ideal word hashing mechanisms, can support future cybersecurity IS studies.

The third essay addresses a key concern held by many CTI professionals and law enforcement officials: "who are the key exploit hackers and exploit sharing communities in online hacker forums?" This essay employs a social network analysis based approach to answer the selected research inquiry. Specifically, an emerging deep learning approach operating on graphs, the Graph Convolutional Network (GCN) was adopted. In this essay, the core operations of the GCN were integrated into the popular deep learning architecture, the autoencoder. The resulting architecture, the Graph Convolutional Autoencoder (GCAE) to automatically generate low-dimensional, latent representations of each hacker in an exploit sharing network based on (1) their connections to other hackers and (2) the exploit content they share with the forum. In addition to the GCAE IT artifact, a novel centrality measure, the Exploit Degree Centrality (EDC) was designed to pinpoint the key hackers on a given exploit sharing network based on the age of their exploit contributions. Although the GCAE and EDC's practical utility were demonstrated with a comprehensive case study of a large English hacker forum, the GCAE artifact can support IS scholars in various SNA related research in selected social media applications.

CTI is a unique domain in that only the most recent information is needed; older exploit knowledge quickly loses value. To aid CTI professionals in this regard, essay IV aims to

answer “what are the emerging hacker exploits?” To answer this question, the fourth essay builds upon the artifact presented in essay III. Specifically, a novel deep learning framework, the Diachronic Graph Convolutional Autoencoder (D-GCAE) was developed. Like essay III, the D-GCAE extends the GCN. Rather than focusing on social networks, however, the GCAE in this essay operates on a text graph representation of hacker forum exploit text by incorporating custom graph convolutions. By constructing multiple time-spells of hacker exploit data and applying the custom GCAE, word embeddings were created. Embedding alignment and semantic displacement measures were implemented on the generated embeddings to map the shifts of exploit terms and identify emerging threats in forums. D-GCAE’s practical value was illustrated with a thorough case study of ransomware. However, the D-GCAE can be viewed as an alternative approach to generating diachronic word embeddings, with the potential of providing a viable alternative to the popular word2vec Neural Network Language Model (NNLM). Consequently, it can aid future IS researchers examining text-based contexts from a methodology with grounding in graph theory.

6.2. Future Research Directions

To date, two dissertations have emerged from IS scholars focused on examining the online hacker community (Benjamin 2016; Li 2017). To the best of my knowledge, this is the first dissertation that explores online hacker forums through the lens of CTI. As with any seminal research, significant opportunities exist for future IS scholars to extend the work presented here to address higher-impact research inquiries. The future directions listed next are by no means exhaustive, nor are they mutually exclusive. Rather, they present an incomplete list of possible opportunities that can open new streams of high-impact CTI research.

(1) Hacker Community Data Fusion: Throughout the course of this dissertation, I argued that hacker forums offer significant CTI value due to its large availability of freely available exploits. However, hacker forums are only one part of the hacker community. Internet-Relay-Chat (IRC), DarkNet Marketplaces (DNMs), and carding shops all remain viable data sources from which proactive CTI can be developed. Each provides its own CTI value: IRC provide real-time chat capabilities and are widely used by hacktivist groups, DarkNet Marketplaces include many leaked databases and new exploits for sale, and carding shops offer numerous sensitive data (e.g., credit cards, Social Security Numbers, etc.) for purchase. Future research can fuse these data sources with hacker forum data to generate novel intelligence. For example, one can develop method(s) to identify when and how exploits from forums transition to DarkNet Markets and vice-versa. Such research pursuits would provide holistic, comprehensive intelligence and understanding of the entire hacker community ecosystem.

(2) Continuous Monitoring Capabilities: Essay IV illustrated the value of identifying emerging threats. CTI is fundamentally a time-sensitive domain; newer exploits provide significantly more value than older. Each essay in this dissertation examined hacker exploits in a batch, fixed manner. However, the dynamic nature of the online hacker community requires an equally robust approach to identifying emerging trends. In light of this, IS scholars can develop novel IT artifacts to facilitate timely, actionable, and thus valuable intelligence. One possibility could be a customized “DarkNet Crawler,” a web spidering system designed specifically to identify new postings on DarkNet Marketplaces and alert relevant stakeholders accordingly.

(3) Cross-Forum/Platform Key Hacker Analysis: As indicated in essay III, identifying key hackers is critical for CTI professionals to attribute cyber-attacks and for law enforcement officials to pursue legal action and/or investigations against selected individuals. As currently presented, extant hacker community literature focuses on identifying key hackers using only one platform. Despite its value, such intelligence provides an incomplete view of a hacker's Tools, Techniques, and Procedures (TTP). To combat this challenge, future studies can identify specific threat actors operating on multiple platforms (forums and otherwise) to gain a complete understanding of their TTP. Possible research avenues would be detecting if they provide specific exploits on one platform compared to others, the timing of their releases, and, in the context of DarkNet Marketplaces, the differences in their pricing strategies for selected products.

7. REFERENCES

- Abbasi, A., Albrecht, C., Vance, A., and Hansen, J. 2012. "Metafraud: A Meta-Learning Framework for Detecting Financial Fraud," *MIS Quarterly* (36:4), JSTOR, pp. 1293–1327.
- Abbasi, A., and Chen, H. 2008. "CyberGate: A Design Framework and System for Text Analysis of Computer-Mediated Communication," *MIS Quarterly* (32:4), JSTOR, pp. 811–837.
- Abbasi, A., Li, W., Benjamin, V., Hu, S., and Chen, H. 2014. "Descriptive Analytics: Examining Expert Hackers in Web Forums," in *Proceedings - 2014 IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014*, pp. 56–63.
(<https://doi.org/10.1109/JISIC.2014.18>).
- Abbasi, A., Zahedi, F. M., Zeng, D., Chen, Y., Chen, H., and Nunamaker, J. F. 2015. "Enhancing Predictive Analytics for Anti-Phishing by Exploiting Website Genre Information," *Journal of Management Information Systems* (31:4), pp. 109–157.
(<https://doi.org/10.1080/07421222.2014.1001260>).
- Abbasi, A., Zhang, Z., Zimbra, D., Chen, H., and Nunamaker Jr, J. F. 2010. "Detecting Fake Websites: The Contribution of Statistical Learning Theory," *MIS Quarterly*, JSTOR, pp. 435–461.
- Ablon, L., Libicki, M. C., and Golay, A. A. 2014. *Markets for Cybercrime Tools and Stolen Data: Hackers' Bazaar*, Rand Corporation.

(http://www.rand.org/content/dam/rand/pubs/research_reports/RR600/RR610/RAND_RR610.pdf).

Abu-El-Haija, S., Kapoor, A., Perozzi, B., and Lee, J. 2018. *N-GCN: Multi-Scale Graph Convolution for Semi-Supervised Node Classification*. (<http://arxiv.org/abs/1802.08888>).

Akimushkin, C., Amancio, D. R., and Oliveira, O. N. 2017. “Text Authorship Identified Using the Dynamics of Word Co-Occurrence Networks,” *PLoS ONE* (12:1). (<https://doi.org/10.1371/journal.pone.0170527>).

Allamanis, M., and Sutton, C. 2013. “Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code,” in *IEEE International Working Conference on Mining Software Repositories*, pp. 53–56. (<https://doi.org/10.1109/MSR.2013.6624004>).

Ayala, L. 2016. *Cybersecurity for Hospitals and Healthcare Facilities*, Berkeley, CA: Apress. (<https://doi.org/10.1007/978-1-4842-2155-6>).

Bajaj, K., Pattabiraman, K., and Mesbah, A. 2014. “Mining Questions Asked by Web Developers,” in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, pp. 112–121. (<https://doi.org/10.1145/2597073.2597083>).

Bakarov, A. 2018. *A Survey of Word Embeddings Evaluation Methods*. (<http://arxiv.org/abs/1801.09536>).

Baldi, P. F., Lopes, C. V., Linstead, E. J., and Bajracharya, S. K. 2008. “A Theory of Aspects as Latent Topics,” *ACM SIGPLAN Notices* (43:10), p. 543.

(<https://doi.org/10.1145/1449955.1449807>).

Barabasi, A. 2016. *Network Science*, Cambridge University Press.

Barua, A., Thomas, S. W., and Hassan, A. E. 2014. “What Are Developers Talking about? An Analysis of Topics and Trends in Stack Overflow,” *Empirical Software Engineering* (19:3), pp. 619–654. (<https://doi.org/10.1007/s10664-012-9231-y>).

Benjamin, V.A. 2016. “Securing Cyberspace: Analyzing Cybercriminal Communities through Web and Text Mining Perspectives” (Doctoral dissertation, University of Arizona).

Benjamin, V. A., and Chen, H. 2013. “Machine learning for attack vector identification in malicious source code,” in *2013 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE pp. 21-23.

Benjamin, V., and Chen, H. 2012. “Securing Cyberspace: Identifying Key Actors in Hacker Communities,” in *ISI 2012 - 2012 IEEE International Conference on Intelligence and Security Informatics: Cyberspace, Border, and Immigration Securities*, pp. 24–29. (<https://doi.org/10.1109/ISI.2012.6283296>).

Benjamin, V., and Chen, H. 2015. “Developing Understanding of Hacker Language through the Use of Lexical Semantics,” in *2015 IEEE International Conference on Intelligence and Security Informatics, ISI 2015*, pp. 79–84. (<https://doi.org/10.1109/ISI.2015.7165943>).

Benjamin, V., Li, W., Holt, T., and Chen, H. 2015. “Exploring Threats and Vulnerabilities in Hacker Web: Forums, IRC and Carding Shops,” in *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, May, pp. 85–90.

(<https://doi.org/10.1109/ISI.2015.7165944>).

Benjamin, V., Zhang, B., Nunamaker, J. F., and Chen, H. 2016. “Examining Hacker Participation Length in Cybercriminal Internet-Relay-Chat Communities,” *Journal of Management Information Systems* (33:2), pp. 482–510.

(<https://doi.org/10.1080/07421222.2016.1205918>).

Blei, D. M., Edu, B. B., Ng, A. Y., Edu, A. S., Jordan, M. I., and Edu, J. B. 2003. “Latent Dirichlet Allocation,” *Journal of Machine Learning Research* (3), pp. 993–1022.

(<https://doi.org/10.1162/jmlr.2003.3.4-5.993>).

Bromiley, M. 2016. “Threat Intelligence: What It Is, and How to Use It Effectively,” *SANS Institute*. (<https://www.sans.org/reading-room/whitepapers/analyst/threat-intelligence-is-effectively-37282>, accessed June 5, 2017).

Caltagirone, S., Pendergast, A., & Betz, C. 2013. *The Diamond Model of Intrusion Analysis*. Center For Cyber Intelligence Analysis and Threat Research Hanover Md.

Cao, S., Lu, W., & Xu, Q. 2015. “Grarep: Learning graph representations with global structural information,” In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* pp. 891-900.

Chen, H., Chiang, R., and Storey, V. 2012. "Business Intelligence and Analytics: From Big Data to Big Impact," *MIS Quarterly* (36:4), pp. 1165-1188.

Chen, J., Ma, T., & Xiao, C. 2018. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling," In *2018 International Conference on Learning Representations*. Forthcoming. *arXiv preprint arXiv:1801.10247*.

Chen, T. H., Thomas, S. W., & Hassan, A. E. 2016. "A survey on the use of topic models when mining software repositories," *Empirical Software Engineering*, (21:5), p. 1843-1919.

Chen, Y. and Zahedi, F. 2016. "Individuals' Internet Security Perceptions and Behaviors: Polycontextual Contrasts Between the United States and China," *MIS Quarterly* (40:1), pp. 205-222.

Chu, B., Holt, T., and Ahn, G. 2010. "Examining the Creation, Distribution, and Function of Malware On Line," *Department of Justice Abstract*, pp. 1-183.

Constantin, L. 2016. "FBI Bought Exploit from Hackers to Access San Bernardino iPhone Computerworld," *Computerworld*.
(<https://www.computerworld.com/article/3055486/security/fbi-bought-exploit-from-hackers-to-access-san-bernardino-iphone.html>, accessed March 25, 2018).

Currim, F., and Ram, S. 2012. "Modeling Spatial and Temporal Set-Based Constraints During Conceptual Database Design," *Information Systems Research* (23:1), pp. 109-128.

Dawande, M., Mookerjee, V., Sriskandarajah, C., and Zhu, Y. 2012. “Structural Search and Optimization in Social Networks,” *INFORMS Journal on Computing* (24:4), pp. 611-623.

Demirel, S. 2017. *Spectral Graph Convolutional Networks for Part-of-Speech Tagging* (Doctoral dissertation, Universität Koblenz-Landau).

El, M., McMahon, E., Samtani, S., Patton, M., and Chen, H. 2017. “Benchmarking Vulnerability Scanners: An Experiment on SCADA Devices and Scientific Instruments,” *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pp. 83 – 88 (<https://doi.org/10.1109/MIS.2018.111145022>)

Elkind, P. 2015. “Sony Pictures: Inside the Hack of the Century,” *Fortune*. (<http://fortune.com/sony-hack-part-1/>, accessed March 25, 2018).

Ernst & Young Global Limited. 2014. “Cyber Threat Intelligence - How to Get ahead of Cybercrime.” ([http://www.ey.com/Publication/vwLUAssets/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime/\\$FILE/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime.pdf](http://www.ey.com/Publication/vwLUAssets/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime/$FILE/EY-cyber-threat-intelligence-how-to-get-ahead-of-cybercrime.pdf), accessed June 5, 2017).

Farnham, G. 2013. “Tools and Standards for Cyber Threat Intelligence Projects,” *SANS Institute*. (<https://www.sans.org/reading-room/whitepapers/warfare/tools-standards-cyber-threat-intelligence-projects-34375>, accessed June 5, 2017).

Faust, K. 1997. “Centrality in Affiliation Networks,” *Social Networks* (19:2), pp. 157–191. ([https://doi.org/10.1016/S0378-8733\(96\)00300-0](https://doi.org/10.1016/S0378-8733(96)00300-0)).

Forum for Incident Response and Security Teams (FIRST). 2017. “Common Vulnerability Scoring System v3.0: Specification Document.” (<https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf>, accessed June 5, 2017).

Friedman, J. 2015. *Definitive Guide to Cyber Threat Intelligence*, CyberEdge Group, LLC. (<https://cryptome.org/2015/09/cti-guide.pdf>).

Fujimoto, K., Williams, M. L., and Ross, M. W. 2013. “Venue-Based Affiliation Networks and HIV Risk-Taking Behavior among Male Sex Workers,” *Sexually Transmitted Diseases* (40:6), pp. 453–458. (<https://doi.org/10.1097/OLQ.0b013e31829186e5>).

Gao, J., Pantel, P., Gamon, M., He, X., and Deng, L. 2014. “Modeling Interestingness with Deep Neural Networks,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2–13.

Ghoshal, A., Menon, S., and Sarkar S. 2015. “Recommendations Using Information from Multiple Association Rules: A Probabilistic Approach,” *Information Systems Research* (26:3), pp. 532–551.

Goel, S. 2011. “Cyberwarfare,” *Communications of the ACM* (54:8), p. 132. (<https://doi.org/10.1145/1978542.1978569>).

Goodfellow, I., Bengio, Y., and Courville. 2016. *Deep Learning*. The MIT Press.

Graham, L. 2017. “Cybercrime costs the global economy \$450 billion”
(<https://www.cnbc.com/2017/02/07/cybercrime-costs-the-global-economy-450-billion-ceo.html>)

Grant, S., Cordy, J. R., and Skillicorn, D. B. 2011. “Reverse Engineering Co-Maintenance Relationships Using Conceptual Analysis of Source Code,” in *2011 18th Working Conference on Reverse Engineering*, pp. 87–91. (<https://doi.org/10.1109/WCRE.2011.20>).

Granville, K. 2015. “9 Recent Cyberattacks Against Big Businesses,” *New York Times*.
(<https://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html>,
accessed March 25, 2018).

Gregor, S., and Hevner, A. R. 2013. “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Quarterly* (37:2), JSTOR, pp. 337-355.

Grisham, J., Samtani, S., Patton, M., and Chen, H. 2017. “Identifying mobile malware and key threat actors in online hacker forums for proactive cyber threat intelligence,” *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pp. 13–18 (<https://doi.org/10.1109/ISI.2017.8004867>).

Grover, A., & Leskovec, J. 2016. “node2vec: Scalable feature learning for networks.” In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, August, pp. 855-864.

Guo, J., Fan, Y., Ai, Q., and Croft, W. B. 2016. “A Deep Relevance Matching Model for Ad-Hoc Retrieval,” in *Proceedings of the 25th ACM International on Conference on*

Information and Knowledge Management - CIKM '16, New York, New York, USA: ACM Press, pp. 55–64. (<https://doi.org/10.1145/2983323.2983769>).

Gupta, A. and Zhdanov, D. 2012. “Growth and Sustainability of Managed Security Services Networks: An Economic Perspective,” *MIS Quarterly* (36:4), pp. 1109-1130.

Hamilton, W. L., Leskovec, J., and Jurafsky, D. 2016. “Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change,” *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing* (2016), pp. 2116–2121. (<http://www.ncbi.nlm.nih.gov/pubmed/28580459>).

Hamilton, W. L., Leskovec, J., and Jurafsky, D. 2016. *Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change*. (<http://arxiv.org/abs/1605.09096>).

Hamilton, W., Ying, Z., & Leskovec, J. 2017. “Inductive representation learning on large graphs,” In *Advances in Neural Information Processing Systems*, pp. 1025-1035.

Hartigan, J. A., and Wong, M. A. 1979. “Algorithm AS 136: A K-Means Clustering Algorithm,” *Applied Statistics* (28:1), p. 100. (<https://doi.org/10.2307/2346830>).

Helpnetsecurity.com. 2009. “January 2009 Threat Scape: Keylogging and Spam Problems, Surge in Exploit Activity,” *Helpnetsecurity.com*. (<https://www.helpnetsecurity.com/2009/02/09/january-2009-threatscape-keylogging-and-spam-problems-surge-in-exploit-activity/>, accessed March 25, 2018).

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1), JSTOR, pp. 75–105.

Higgins, K. J. 2014. "SQL Injection Attacks Haunt Retailers," *DarkReading.com*. (<https://www.darkreading.com/sql-injection-attacks-haunt-retailers/d/d-id/1269576>, accessed March 25, 2018).

Holt, T. J. 2013. "Examining the Forces Shaping Cybercrime Markets Online," *Social Science Computer Review* (31:2), pp. 165–177. (<https://doi.org/10.1177/0894439312452998>).

Holt, T. J., Strumsky, D., Smirnova, O., and Kilger, M. 2012. "Examining the Social Networks of Malware Writers and Hackers," (6:1), pp. 891–903.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. 2013. "Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data," in *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management - CIKM '13*, New York, New York, USA: ACM Press, pp. 2333–2338. (<https://doi.org/10.1145/2505515.2505665>).

Huang, S., and Chen, H. 2016. "Exploring the Online Underground Marketplaces through Topic-Based Social Network and Clustering," in *2016 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 145–150.

Hui, K.L., Vance, A., Zhdanov, D. "Securing Digital Assets," in *MIS Quarterly Research Curations*, Ashley Bush, ed., <http://misq.org/research-curations>, May 27, 2016.

- Hutchings, A., and Holt, T. J. 2015. "A Crime Script Analysis of the Online Stolen Data Market," *British Journal of Criminology* (55:3), pp. 596–614.
(<https://doi.org/10.1093/bjc/azu106>).
- Jaech, A., Kamisetty, H., Ringger, E., and Clarke, C. 2017. *Match-Tensor: A Deep Relevance Model for Search*. (<http://arxiv.org/abs/1701.07795>).
- Jia, C., Li, Y., Carson, M. B., Wang, X., & Yu, J. 2017. Node Attribute-enhanced Community Detection in Complex Networks. *Scientific Reports*, (7:1), 2626.
- Keegan, B., Gergle, D., and Contractor, N. 2012. "Do Editors or Articles Drive Collaboration?," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work - CSCW '12*, p. 427. (<https://doi.org/10.1145/2145204.2145271>).
- Kennedy, D., O’Gorman, J., Kearns, D., and Aharoni, M. 2011. *Metasploit: The Penetration Tester’s Guide*, (1st edition), No Starch Press.
- Ketter, W., Peters, M., Collins, J., and Gupta, A. 2016. "A Multiagent Competitive Gaming Platform to Address Societal Challenges," *MIS Quarterly* (40:2), pp. 447-460.
- Kim, S., and Kim, B. 2014. "Differential Effects of Prior Experience on the Malware Resolution Process," *MIS Quarterly* (38:3), pp. 655-678.
- Kime, B. P. 2016. "Threat Intelligence: Planning and Direction," *SANS Institute*. (<https://www.sans.org/reading-room/whitepapers/threats/threat-intelligence-planning-direction-36857>, accessed June 5, 2017).

Kipf, T. N., & Welling, M. 2016. “Variational graph auto-encoders,” in *Neural Information Processing Systems Bayesian Deep Learning Workshop*.

Kipf, T. N., & Welling, M. 2017. “Semi-supervised classification with graph convolutional networks,” In *International Conference on Learning Representations*, *arXiv preprint*. (<http://arxiv.org/abs/1609.02907>).

Kitten, T. 2014. “Target Malware: Exploring the Origins.” (<http://www.bankinfosecurity.com/interviews/intelcrawler-i-2161>, accessed June 5, 2017).

Kwon, J. and Johnson, M. 2014. “Proactive Versus Reactive Security Investments in the Healthcare Sector,” *MIS Quarterly* (38:2), pp. 451-471.

L’Huillier, G., Ríos, S. A., Alvarez, H., and Aguilera, F. 2010. “Topic-Based Social Network Analysis for Virtual Communities of Interests in the Dark Web,” in *ACM SIGKDD Workshop on Intelligence and Security Informatics - ISI-KDD ’10*, pp. 1–9. (<https://doi.org/10.1145/1938606.1938615>).

LeCun, Y., Bengio, Y., and Hinton, G. 2015. “Deep Learning,” *Nature* (521:7553), pp. 436–444. (<https://doi.org/10.1038/nature14539>).

Lee, M. and Lee, M. 2017. “The Hunter Strikes Back: The 2017 Threat Hunting Survey,” *SANS Institute*, (<https://www.sans.org/reading-room/whitepapers/analyst/hunter-strikes-back-2017-threat-hunting-survey-37760>, accessed January 11, 2018)

Levine, M., and Date, J. 2015. "22 Million Affected by OPM Hack, Officials Say," *ABC News*.

(<http://abcnews.go.com/US/exclusive-25-million-affected-opm-hack-sources/story?id=32332731>, accessed March 25, 2018).

Li, C., Peters, G., Richardson, V., and Watson, M. "The Consequences of Information Technology Control Weaknesses on Management Information Systems: The Case of Sarbanes-Oxley Internal Control Reports," *MIS Quarterly* (36:1), pp. 179-203.

Li, W. 2017. "Towards Secure and Trustworthy Cyberspace: Social Media Analytics on Hacker Communities" (Doctoral dissertation, University of Arizona).

Li, W., and Chen, H. 2014. "Identifying Top Sellers In Underground Economy Using Deep Learning-Based Sentiment Analysis," in *2014 IEEE Joint Intelligence and Security Informatics Conference*, IEEE, September, pp. 64–67.

(<https://doi.org/10.1109/JISIC.2014.19>).

Li, W., Chen, H., and Nunamaker, J. F. 2016. "Identifying and Profiling Key Sellers in Cyber Carding Community: AZSecure Text Mining System," *Journal of Management Information Systems* (33:4), pp. 1059–1086.

(<https://doi.org/10.1080/07421222.2016.1267528>).

Li, W., Yin, J., and Chen, H. 2016. "Targeting Key Data Breach Services in Underground Supply Chain," in *IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016*, pp. 322–324.

(<https://doi.org/10.1109/ISI.2016.7745501>).

Lin, Y.-K., Chen, H., Brown, R. A., Li, S.-H., and Yang, H.-J. 2017. "Healthcare Predictive Analytics For Risk Profiling In Chronic Care: A Bayesian Multitask Learning Approach," *MIS Quarterly* (41:2), JSTOR, pp. 473-495.

Linares-Vásquez, M., McMillan, C., Poshyvanyk, D., and Grechanik, M. 2014. "On Using Machine Learning to Automatically Classify Software Applications into Domain Categories," *Empirical Software Engineering* (19:3), pp. 582–618.

(<https://doi.org/10.1007/s10664-012-9230-z>).

Linstead, E., Lopes, C., and Baldi, P. 2008. "An Application of Latent Dirichlet Allocation to Analyzing Software Evolution," in *Proceedings - 7th International Conference on Machine Learning and Applications, ICMLA 2008*, pp. 813–818.

(<https://doi.org/10.1109/ICMLA.2008.47>).

Liska, A. 2014. *Building an Intelligence-Led Security Program*, Syngress.

MacDonald, M., & Frank, R. 2017. The network structure of malware development, deployment and distribution. *Global Crime*, (18:1), pp. 49-69.

- Mahmood, M. A., Siponen, M., Straub, D., Rao, H. R., and Raghu, T. S. 2010. "Moving toward Black Hat Research in Information Systems Security: An Editorial Introduction to the Special Issue," *MIS Quarterly* (34:3), JSTOR, pp. 431–433.
- Manessi, F., Rozza, A., and Manzo, M. 2017. "Dynamic Graph Convolutional Networks," *arXiv*, pp. 1–16 (available at <http://arxiv.org/abs/1704.06199>).
- Marin, E., Diab, A., and Shakarian, P. 2016. "Product Offerings in Malicious Hacker Markets," in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 187–189. (<https://doi.org/10.1109/ISI.2016.7745465>).
- Maskeri, G., Sarkar, S., and Heafield, K. 2008. "Mining Business Topics in Source Code Using Latent Dirichlet Allocation," in *Proceedings of the 1st India Software Engineering Conference (ISEC '08)*, pp. 113–120. (<https://doi.org/10.1145/1342211.1342234>).
- Mathews, L. 2016. "World's Biggest Mirai Botnet Is Being Rented Out For DDoS Attacks," *Forbes*. (<https://www.forbes.com/sites/leemathews/2016/11/29/worlds-biggest-mirai-botnet-is-being-rented-out-for-ddos-attacks/#f54f54f58ad6>, accessed June 5, 2017)
- McMahon, E., Williams, R., El, M., Samtani, S., Patton, M., and Chen, H. 2017. "Assessing medical device vulnerabilities on the Internet of Things," *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pp. 176 – 178 (<https://doi.org/10.1109/ISI.2017.8004903>).

- McMillan, C., Linares-Vásquez, M., Poshyvanyk, D., and Grechanik, M. 2011. “Categorizing Software Applications for Maintenance,” in *IEEE International Conference on Software Maintenance, ICSM*, pp. 343–352.
(<https://doi.org/10.1109/ICSM.2011.6080801>).
- McMillan, R. 2012. “How the Boy Next Door Accidentally Built a Syrian Spy Tool,” *Wired.com*. (<https://www.wired.com/2012/07/dark-comet-syrian-spy-tool/>, accessed March 25, 2018).
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. “Distributed Representations of Words and Phrases and Their Compositionality,” in *NIPS*, pp. 1–9.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. “Efficient Estimation of Word Representations in Vector Space,” *arXiv Preprint arXiv:1301.3781*, pp. 1–12.
(<https://doi.org/10.1162/153244303322533223>).
- Mitra, B., Diaz, F., and Craswell, N. 2016. *Learning to Match Using Local and Distributed Representations of Text for Web Search*. (<http://arxiv.org/abs/1610.08136>).
- Motoyama, M., McCoy, D., Levchenko, K., Savage, S., and Voelker, G. M. 2011. “An Analysis of Underground Forums,” in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference - IMC '11*, p. 71.
(<https://doi.org/10.1145/2068816.2068824>).
- Nastase, V., Mihalcea, R., and Radev, D. R. 2015. “A survey of graphs in natural language processing,” *Natural Language Engineering*, (21:5), 665-698.

National Science and Technology Council (NSTC). 2011. “Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity Research and Development Program,” *Executive Office of the President, National Science and Technology Council*.

Nunamaker Jr, J. F., Chen, M., and Purdin, T. D. M. 1990. “Systems Development in Information Systems Research,” *Journal of Management Information Systems* (7:3), Taylor & Francis, pp. 89–106.

Nunamaker, J. F., Briggs, R. O., Derrick, D. C., and Schwabe, G. 2015. “The Last Research Mile: Achieving Both Rigor and Relevance in Information Systems Research,” *Journal of Management Information Systems* (32:3), pp. 10–47.
(<https://doi.org/10.1080/07421222.2015.1094961>).

Nunamaker, J. F., Twyman, N. W., Giboney, J. S., and Briggs, R. O. 2017. “Creating High-Value Real-World Impact through Systematic Programs of Research,” *MIS Quarterly* (41:2), pp. 335–351. (<https://doi.org/10.25300/MISQ/2017/41.2.01>).

Nunes, E., Diab, A., Gunn, A., Marin, E., Mishra, V., Paliath, V., Robertson, J., Shakarian, J., Thart, A., and Shakarian, P. 2016. “Darknet and Deepnet Mining for Proactive Cybersecurity Threat Intelligence,” in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 7–12. (<https://doi.org/10.1109/ISI.2016.7745435>).

Pang, L., Lan, Y., Guo, J., Xu, J., and Cheng, X. 2016. *A Study of MatchPyramid Models on Ad-Hoc Retrieval*. (<http://arxiv.org/abs/1606.04648>).

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. 2007. “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems* (24:3), Taylor & Francis, pp. 45–77.

Perozzi, B., Al-Rfou, R., and Skiena, S. 2014. “Deepwalk: Online learning of social representations,” In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 701-710.

Prat, N., Comyn-Wattiau, I., and Akoka, J. 2015. “A Taxonomy of Evaluation Methods for Information Systems Artifacts,” *Journal of Management Information Systems* (32:3), pp. 229–267. (<https://doi.org/10.1080/07421222.2015.1099390>).

Radianti, J. 2010. “A Study of a Social Behavior inside the Online Black Markets,” in *Proceedings - 4th International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2010*, pp. 189–194. (<https://doi.org/10.1109/SECURWARE.2010.38>).

Rai, A. 2017. “Editor’s Comments: Diversity of Design Science Research,” *MIS Quarterly*, (41:1), pp. iii– xviii.

Randbotham, S., Mitra, S., and Ramsey, J. “Are Markets for Vulnerabilities Effective?” *MIS Quarterly*, (36:1), pp. 43-64.

Riley, M., Elgin, B., Lawrence, D., and Matlack, C. 2014. “Target Missed Warnings in Epic Hack of Credit Card Data,” *Bloomberg*.

(<https://www.bloomberg.com/news/articles/2014-03-13/target-missed-warnings-in-epic-hack-of-credit-card-data>, accessed March 25, 2018).

Rios, S., Aguilera, F., Bustos, F., Omitola, T., and Shadbolt, N. 2011. “Leveraging Social Network Analysis with Topic Models and the Semantic Web,” *2011 IEEE WICACM International Conferences on Web Intelligence and Intelligent Agent Technology* (3:5), pp. 339–342. (<https://doi.org/10.1109/WI-IAT.2011.127>).

Robertson, J., Diab, A., Marin, E., Nunes, E., Paliath, V., Shakarian, J., and Shakarian, P. 2017. *Darkweb Cyber Threat Intelligence Mining*, Cambridge: Cambridge University Press.

Rosenberg, A., and Hirschberg, J. 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 (EMNLP-CoNLL)*, June, pp. 410-420.

Rousseau, F., Kiagias, E., and Vazirgiannis, M. 2015. “Text Categorization as a Graph Classification Problem,” *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1702–1712. (<http://www.aclweb.org/anthology/P15-1164>).

Samtani, S., Yu, S., Zhu, H., Patton, M., Matherly, J., and Chen, H. 2018. “Identifying SCADA Systems and their Vulnerabilities on the Internet of Things: A Text Mining Approach,” *IEEE Intelligent Systems* (33:2), pp. 63-73.

(<https://10.1109/MIS.2018.111145022>).

Samtani, S. and Chen, H. 2016. “AZSecure Hacker Assets Portal: Cyber Threat Intelligence and Malware Analysis,” in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 319–321.

(<https://doi.org/10.1109/ISI.2016.7745500>).

Samtani, S., Chinn, K., Larson, C., and Chen, H. 2016. “AZSecure Hacker Assets Portal: Cyber Threat Intelligence and Malware Analysis,” in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 19–24.

(<https://doi.org/10.1109/ISI.2016.7745437>).

Samtani, S., Chinn, R., and Chen, H. 2015. “Exploring Hacker Assets in Underground Forums,” in *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, May, pp. 31–36. (<https://doi.org/10.1109/ISI.2015.7165935>).

Samtani, S., Chinn, R., Chen, H., and Nunamaker, J. F. 2017. “Exploring Emerging Hacker Assets and Key Hackers for Proactive Cyber Threat Intelligence,” *Journal of Management Information Systems* (34:4), pp. 1023–1053.

(<https://doi.org/10.1080/07421222.2017.1394049>).

Samtani, S., Yu, S., Zhu, H., Patton, M., and Chen, H. 2016. “Identifying SCADA Vulnerabilities Using Passive and Active Vulnerability Assessment Techniques,” in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, September, pp. 25–30. (<https://doi.org/10.1109/ISI.2016.7745438>).

Sandle, P., and Char, P. 2014. “Cyber Crime Costs Global Economy \$445 Billion a Year: Report,” *Reuters*. (<https://www.reuters.com/article/us-cybersecurity-mcafee-csis/cyber-crime-costs-global-economy-445-billion-a-year-report-idUSKBN0EK0SV20140609>, accessed March 25, 2018).

Savage, T., Dit, B., Gethers, M., and Poshyvanyk, D. 2010. “Topic<inf>XP</inf>: Exploring Topics in Source Code Using Latent Dirichlet Allocation,” *2010 IEEE International Conference on Software Maintenance*, pp. 1–6. (<https://doi.org/10.1109/ICSM.2010.5609654>).

Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. 2015. “Evaluation Methods for Unsupervised Word Embeddings,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 298–307. (<https://doi.org/10.18653/v1/D15-1036>).

Schönemann, P. H. 1966. “A Generalized Solution of the Orthogonal Procrustes Problem,” *Psychometrika* (31:1), pp. 1–10. (<https://doi.org/10.1007/BF02289451>).

Shackleford, D. 2015. “Who’s Using Cyberthreat Intelligence and How?” *SANS Institute*. (<https://www.sans.org/reading-room/whitepapers/analyst/cyberthreat-intelligence-how-35767>, accessed June 5, 2017).

Shackleford, D. 2016. “2016 Security Analytics Survey,” *SANS Institute*. (<https://www.sans.org/reading-room/whitepapers/analyst/2016-security-analytics-survey-37467>, accessed June 5, 2017).

Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. 2014. “Learning Semantic Representations Using Convolutional Neural Networks for Web Search,” in *Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion*, New York, New York, USA: ACM Press, pp. 373–374.

(<https://doi.org/10.1145/2567948.2577348>).

Song, Y., Elkahky, A. M., and He, X. 2016. “Multi-Rate Deep Learning for Temporal Recommendation,” in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '16*, New York, New York, USA: ACM Press, pp. 909–912. (<https://doi.org/10.1145/2911451.2914726>).

Sood, A. K., and Enbody, R. J. 2013. “Crimeware-as-a-service—A Survey of Commoditized Crimeware in the Underground Market,” *International Journal of Critical Infrastructure Protection* (6:1), pp. 28–38. (<https://doi.org/10.1016/j.ijcip.2013.01.002>).

Spears, J. and Barki, H. “User Participation in Information Systems Security Risk Management,” *MIS Quarterly* (34:3), pp. 503-522.

Stewart, S. A., and Abidi, S. S. R. 2012. “Applying Social Network Analysis to Understand the Knowledge Sharing Behaviour of Practitioners in a Clinical Online Discussion Forum,” *Journal of Medical Internet Research* (14:6), pp. 245–264.

(<https://doi.org/10.2196/jmir.1982>).

Stuart, K. 2015. “Lizard Squad Is Back: Group ‘Attacks Xbox Live and Daybreak Games,’” *The Guardian*. (<https://www.theguardian.com/technology/2015/feb/16/lizard-squad-attacks-xbox-live-daybreak-games>, accessed March 25, 2018).

Symantec Corporation. 2014. "Internet Security Threat Report," (Vol. 19). (http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf).

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. 2015. "Line: Large-scale information network embedding," In *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, May, pp. 1067-1077.

Tian, K., Reville, M., and Poshyanyk, D. 2009. "Using Latent Dirichlet Allocation for Automatic Categorization of Software," in *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, MSR 2009*, pp. 163–166. (<https://doi.org/10.1109/MSR.2009.5069496>).

U.S. News & World Report. 2016. "U.S. News & World Report Announces the 2016–17 Best Hospitals." (<https://www.usnews.com/info/blogs/press-room/articles/2016-08-02/us-news-announces-the-201617-best-hospitals>, accessed June 5, 2017).

Ugurel, S., Krovetz, R., 'Z, C. L. G., Pennock, D. M., Glover, E. J., and Zha, H. 2002. "What's the Code? Automatic Classification of Source Code Archives," in *Proceedings of the Eighth Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*(Vol. 2), pp. 632–638. (<https://doi.org/http://doi.acm.org/10.1145/775047.775141>).

Vance, A., Lowry, P., and Eggett, D. 2015. "Increasing Accountability Through User-Interface Design Artifacts: A New Approach to Addressing the Problem of Access Policy Violations," *MIS Quarterly* (39:2), pp. 345-366.

- Vijayan, J. 2014. “New Details Of Home Depot Attack Reminiscent Of Target’s Breach,” *DarkReading.com*. (<https://www.darkreading.com/attacks-breaches/new-details-of-home-depot-attack-reminiscent-of-targets-breach/d/d-id/1317323>, accessed March 25, 2018).
- Wang, J., Gupta, M., and Rao, H. 2015. “Insider Threats in a Financial Institution: Analysis of Attack-Proneess of Information Systems Applications,” *MIS Quarterly* (39:1), pp 91-112.
- Wang, T., Wang, H., Yin, G., Ling, C. X., Li, X., and Zou, P. 2013. “Mining Software Profile across Multiple Repositories for Hierarchical Categorization,” in *IEEE International Conference on Software Maintenance, ICSM*, pp. 240–249. (<https://doi.org/10.1109/ICSM.2013.35>).
- Weidman, G. 2014. *Penetration Testing: A Hands-On Introduction to Hacking*, (1st ed.), No Starch Press. (<https://www.nostarch.com/pentesting>).
- Williams, R., McMahon, E., Samtani, S., Patton, M., and Chen, H. 2017. “Identifying Vulnerabilities of Consumer Internet of Things (IoT) devices: A scalable approach,” *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pp. 179 – 181 (<https://doi.org/10.1109/ISI.2017.8004904>).
- Wood, S. 2009. “How to Find a Backdoor in a Hacked WordPress,” *Ottopress.com*. (<http://ottopress.com/2009/hacked-wordpress-backdoors/>, accessed March 25, 2018).
- Yan, S., Xiong, Y., and Lin, D. 2018. *Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition*. (<http://arxiv.org/abs/1801.07455>).

- Yao, Z., Sun, Y., Ding, W., Rao, N., and Xiong, H. 2018. “Dynamic Word Embeddings for Evolving Semantic Discovery,” in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18*, New York, New York, USA: ACM Press, pp. 673–681. (<https://doi.org/10.1145/3159652.3159703>).
- Yasunaga, M., Zhang, R., Meelu, K., Pareek, A., Srinivasan, K., and Radev, D. 2017. *Graph-Based Neural Multi-Document Summarization*. (<http://arxiv.org/abs/1706.06681>).
- Ye, X., Qi, Z., and Massey, D. 2015. “Learning Relevance from Click Data via Neural Network Based Similarity Models,” in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, October, pp. 801–806. (<https://doi.org/10.1109/BigData.2015.7363825>).
- Yip, M. 2011. “An Investigation into Chinese Cybercrime and the Applicability of Social Network Analysis,” in *Proceedings of the ACM WebSci'11*, pp. 1–4. (<http://eprints.ecs.soton.ac.uk/22500/>).
- Zeng, D., Chen, H., Lusch, R., and Li, S.-H. 2010. “Social Media Analytics and Intelligence,” *IEEE Intelligent Systems* (25:6), pp. 13–16. (<https://doi.org/10.1109/MIS.2010.151>).
- Zhai, M., Tan, J., and Choi, J. D. 2016. “Intrinsic and Extrinsic Evaluations of Word Embeddings,” *The 30th AAAI Conference on Artificial Intelligence*, pp. 4282–4283. (<http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12454/12257>).

Zhang, Y., Rahman, M. M., Braylan, A., Dang, B., Chang, H.-L., Kim, H., McNamara, Q., Angert, A., Banner, E., Khetan, V., McDonnell, T., Nguyen, A. T., Xu, D., Wallace, B. C., and Lease, M. 2016. *Neural Information Retrieval: A Literature Review*.

(<http://arxiv.org/abs/1611.06792>).

Zhao, Z., Sankaran, M., Ahn, G.-J., Holt, T. J., Jing, Y., and Hu, H. 2016. “Mules, Seals, and Attacking Tools: Analyzing 12 Online Marketplaces,” *IEEE Security & Privacy* (14:3), pp. 32–43. (<https://doi.org/10.1109/MSP.2016.46>).